US011182486B2

(12) **United States Patent**
**Cosgrove et al.**

(10) **Patent No.: US 11,182,486 B2**
(45) **Date of Patent: Nov. 23, 2021**

(54) **EARLY BOOT DRIVER FOR START-UP DETECTION OF MALICIOUS CODE**

(71) Applicant: **Sophos Limited**, Abingdon (GB)

(72) Inventors: **Richard Paul Cosgrove**, Abingdon (GB); **Mark David Harris**, Oxon (GB); **Andrew G. P. Smith**, Oxford (GB)

(73) Assignee: **Sophos Limited**, Abingdon (GB)

( * ) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 151 days.

(21) Appl. No.: **16/438,045**

(22) Filed: **Jun. 11, 2019**

(65) **Prior Publication Data**

US 2020/0394305 A1 Dec. 17, 2020

(51) **Int. Cl.**

| | |
|---|---|
| ***G06F 21/00*** | (2013.01) |
| ***G06F 21/57*** | (2013.01) |
| ***G06F 21/54*** | (2013.01) |
| ***G06F 21/56*** | (2013.01) |

(52) **U.S. Cl.**

CPC ..... ***G06F 21/575*** (2013.01); ***G06F 21/54*** (2013.01); ***G06F 21/56*** (2013.01); *G06F 2221/033* (2013.01)

(58) **Field of Classification Search**

None

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS


| | | | | |
|---|---|---|---|---|
| 7,783,886 | B2 * | 8/2010 | Walmsley | G06F 21/64 713/176 |
| 7,934,261 | B1 | 4/2011 | Tan et al. | |
| 8,856,927 | B1 * | 10/2014 | Beloussov | G06F 21/53 726/23 |
| 9,336,385 | B1 * | 5/2016 | Spencer | G06F 21/56 |
| 2004/0181303 | A1 * | 9/2004 | Walmsley | B41J 2/04541 700/115 |
| 2006/0236108 | A1 | 10/2006 | Andrews | |
| 2008/0177994 | A1 | 7/2008 | Mayer | |
| 2012/0060217 | A1 | 3/2012 | Sallam | |
| 2012/0254982 | A1 * | 10/2012 | Sallam | G06F 21/564 726/16 |
| 2013/0347131 | A1 * | 12/2013 | Mooring | G06F 21/50 726/29 |
| 2017/0147819 | A1 * | 5/2017 | Vasilenko | G06F 21/554 |
| 2019/0205533 | A1 * | 7/2019 | Diehl | H04L 9/3268 |

(Continued)

OTHER PUBLICATIONS

Gu, Zhongshu et al. DRIP: A framework for purifying trojaned kernel drivers. 2013 43rd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN). https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6575342 (Year: 2013).*

(Continued)

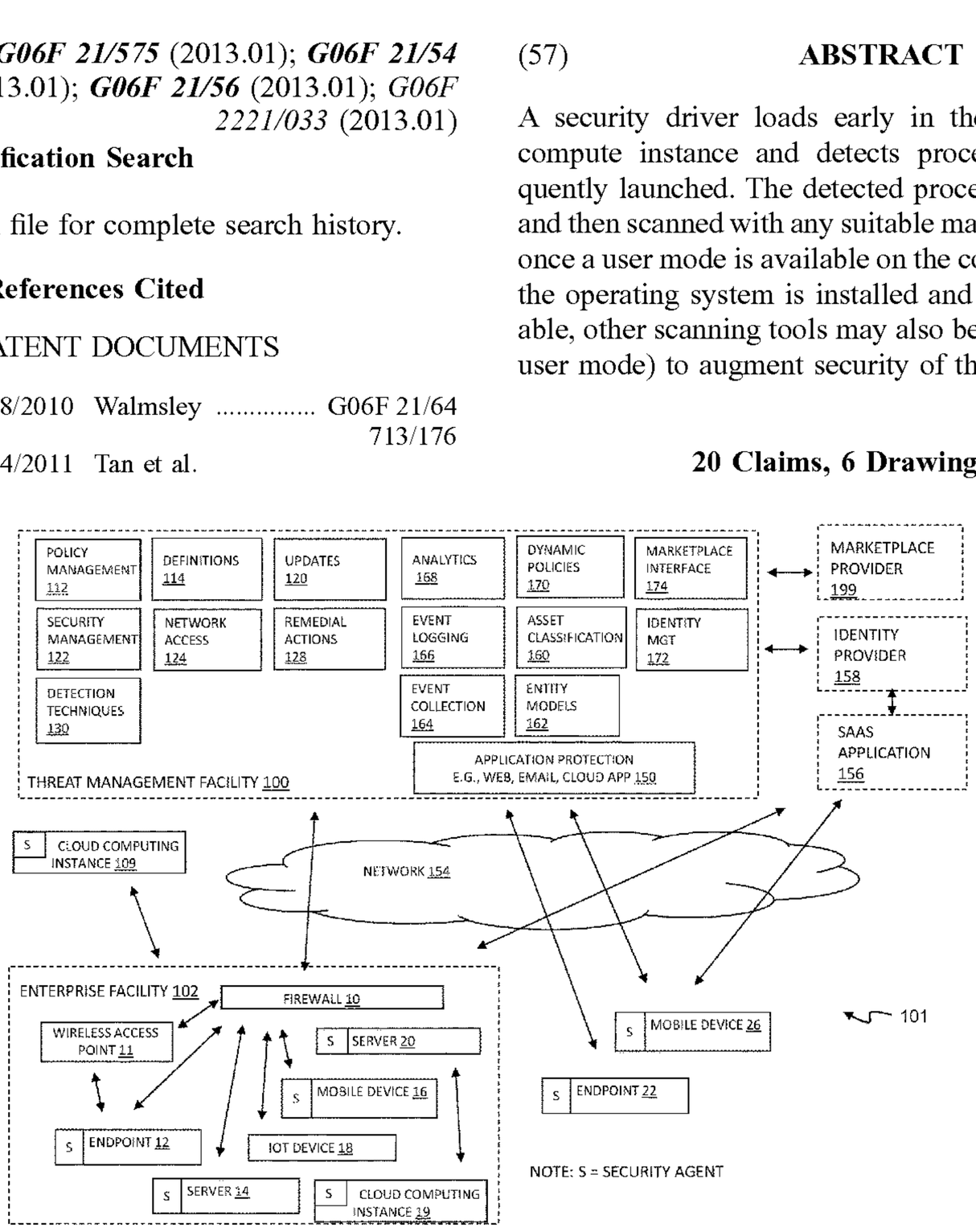

*Primary Examiner* — Jeremiah L Avery

(74) *Attorney, Agent, or Firm* — Strategic Patents, P.C.

(57) **ABSTRACT**

A security driver loads early in the boot process for a compute instance and detects processes that are subsequently launched. The detected processes can be recorded, and then scanned with any suitable malware scanning tool(s) once a user mode is available on the compute instance. After the operating system is installed and a user mode is available, other scanning tools may also be deployed (e.g., in the user mode) to augment security of the compute instance.

**20 Claims, 6 Drawing Sheets**



(56) **References Cited**

U.S. PATENT DOCUMENTS

2020/0394300 A1 12/2020 Harris et al.

OTHER PUBLICATIONS

Zhang, Cong et al. Research and implementation of file security mechanisms based on file system filter driver. 2017 Annual Reliability and Maintainability Symposium (RAMS). https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7889772 (Year: 2017).*
Musavi, Seyyedeh Atefeh; Kharrazi, Mehdi. Back to Static Analysis for Kernel-Level Rootkit Detection. IEEE Transactions on Information Forensics and Security, vol. 9, Issue: 9. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6850033 (Year: 2014).*
Rhee, Junghwan. Data-Centric OS Kernel Malware Characterization. IEEE Transactions on Information Forensics and Security vol. 9, Issue: 1. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6671356 (Year: 2014).*
Li, Shaobo et al. Research and Application of USB Filter Driver Based on Windows Kernel. 2010 Third International Symposium on Intelligent Information Technology and Security Informatics, https://ieeexplore.IEEE.org/stamp/stamp.jsp?tp=&arnumber=5453612 (Year: 2010).*
Javaheri, Danial et al. Detection and Elimination of Spyware and Ransomware by Intercepting Kernel-Level System Routines. IEEE Access, vol. 6. https://ieeexplore.IEEE.org/stamp/stamp.jsp?tp=&arnumber=8566151 (Year: 2018).*
Kirmani, Mariya S et al., "Analyzing Detection Avoidance of Malware by Process Hiding", 2018, 5 pages.
Shahzad, Farrukh et al., "In-Execution Malware Detection using Task Structures of Linux Processes", 2011, 6 pages.
USPTO, "U.S. Appl. No. 16/437,999 Notice of Allowance dated Dec. 9, 2020", 15 pages.

* cited by examiner

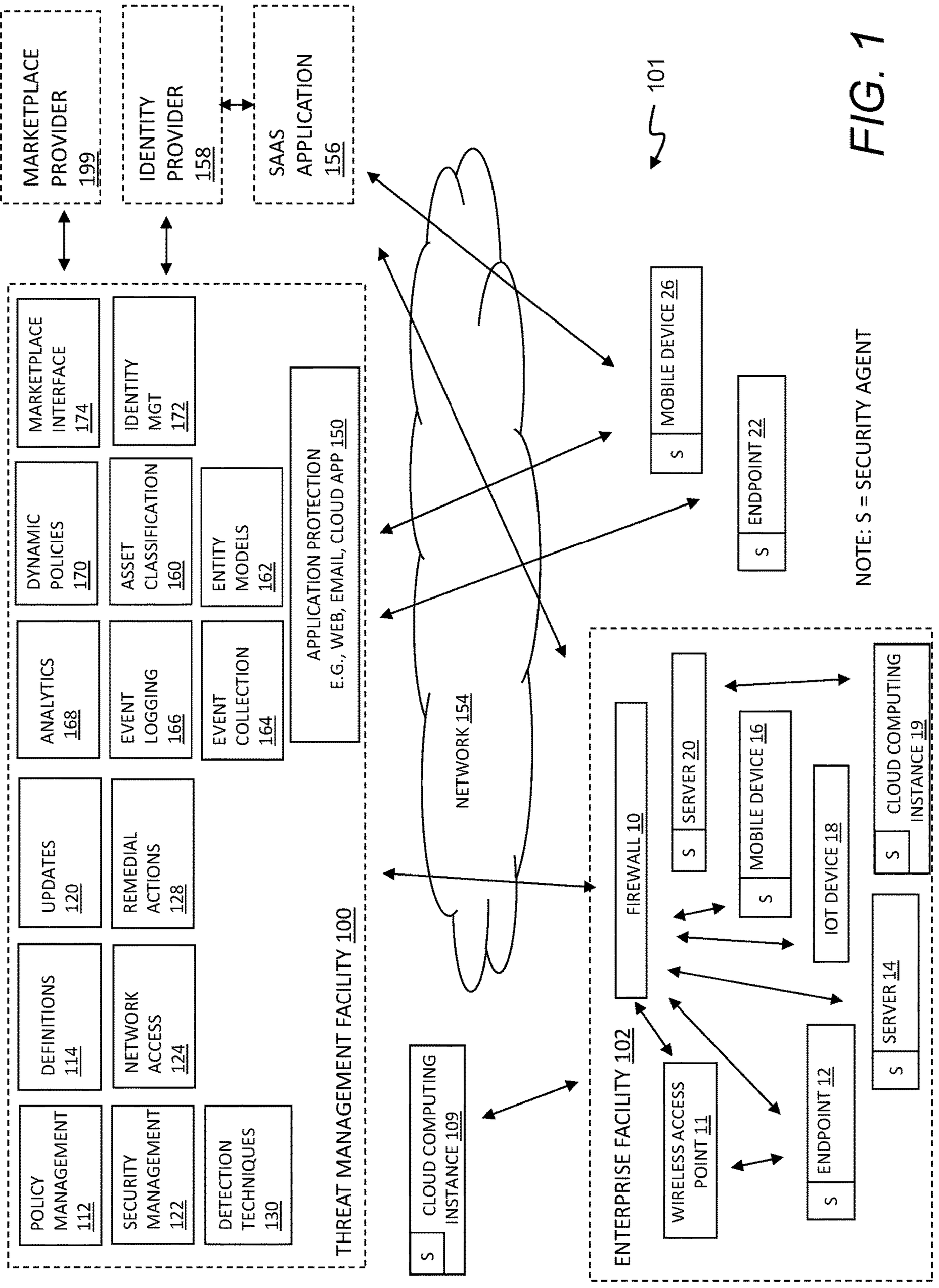

MARKETPLACE PROVIDER 199
IDENTITY PROVIDER 158
SAAS APPLICATION 156
101
MARKETPLACE INTERFACE 174
IDENTITY MGT 172
DYNAMIC POLICIES 170
ASSET CLASSIFICATION 160
ENTITY MODELS 162
ANALYTICS 168
EVENT LOGGING 166
EVENT COLLECTION 164
APPLICATION PROTECTION E.G., WEB, EMAIL, CLOUD APP 150
UPDATES 120
REMEDIAL ACTIONS 128
DEFINITIONS 114
NETWORK ACCESS 124
POLICY MANAGEMENT 112
SECURITY MANAGEMENT 122
DETECTION TECHNIQUES 130
THREAT MANAGEMENT FACILITY 100
NETWORK 154
S CLOUD COMPUTING INSTANCE 109
S MOBILE DEVICE 26
S ENDPOINT 22
NOTE: S = SECURITY AGENT
ENTERPRISE FACILITY 102
FIREWALL 10
S SERVER 20
S MOBILE DEVICE 16
S CLOUD COMPUTING INSTANCE 19
IOT DEVICE 18
WIRELESS ACCESS POINT 11
S ENDPOINT 12
S SERVER 14

FIG. 1

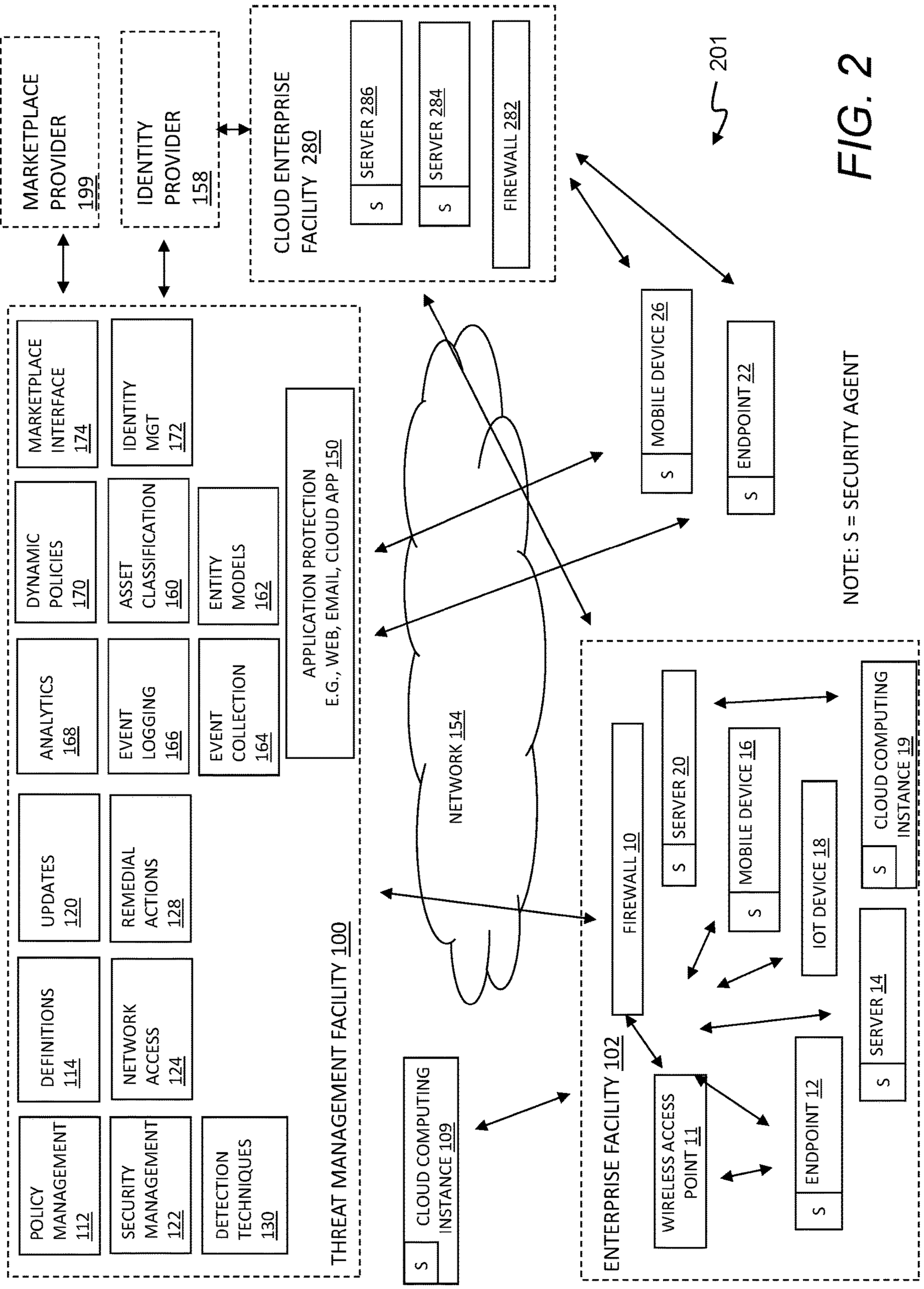

MARKETPLACE PROVIDER 199
IDENTITY PROVIDER 158
CLOUD ENTERPRISE FACILITY 280
S SERVER 286
S SERVER 284
FIREWALL 282
201
MARKETPLACE INTERFACE 174
IDENTITY MGT 172
DYNAMIC POLICIES 170
ASSET CLASSIFICATION 160
ENTITY MODELS 162
ANALYTICS 168
EVENT LOGGING 166
EVENT COLLECTION 164
APPLICATION PROTECTION E.G., WEB, EMAIL, CLOUD APP 150
UPDATES 120
REMEDIAL ACTIONS 128
DEFINITIONS 114
NETWORK ACCESS 124
POLICY MANAGEMENT 112
SECURITY MANAGEMENT 122
DETECTION TECHNIQUES 130
THREAT MANAGEMENT FACILITY 100
NETWORK 154
S MOBILE DEVICE 26
S ENDPOINT 22
NOTE: S = SECURITY AGENT
S CLOUD COMPUTING INSTANCE 109
ENTERPRISE FACILITY 102
FIREWALL 10
S SERVER 20
S MOBILE DEVICE 16
IOT DEVICE 18
S CLOUD COMPUTING INSTANCE 19
WIRELESS ACCESS POINT 11
S ENDPOINT 12
S SERVER 14

FIG. 2

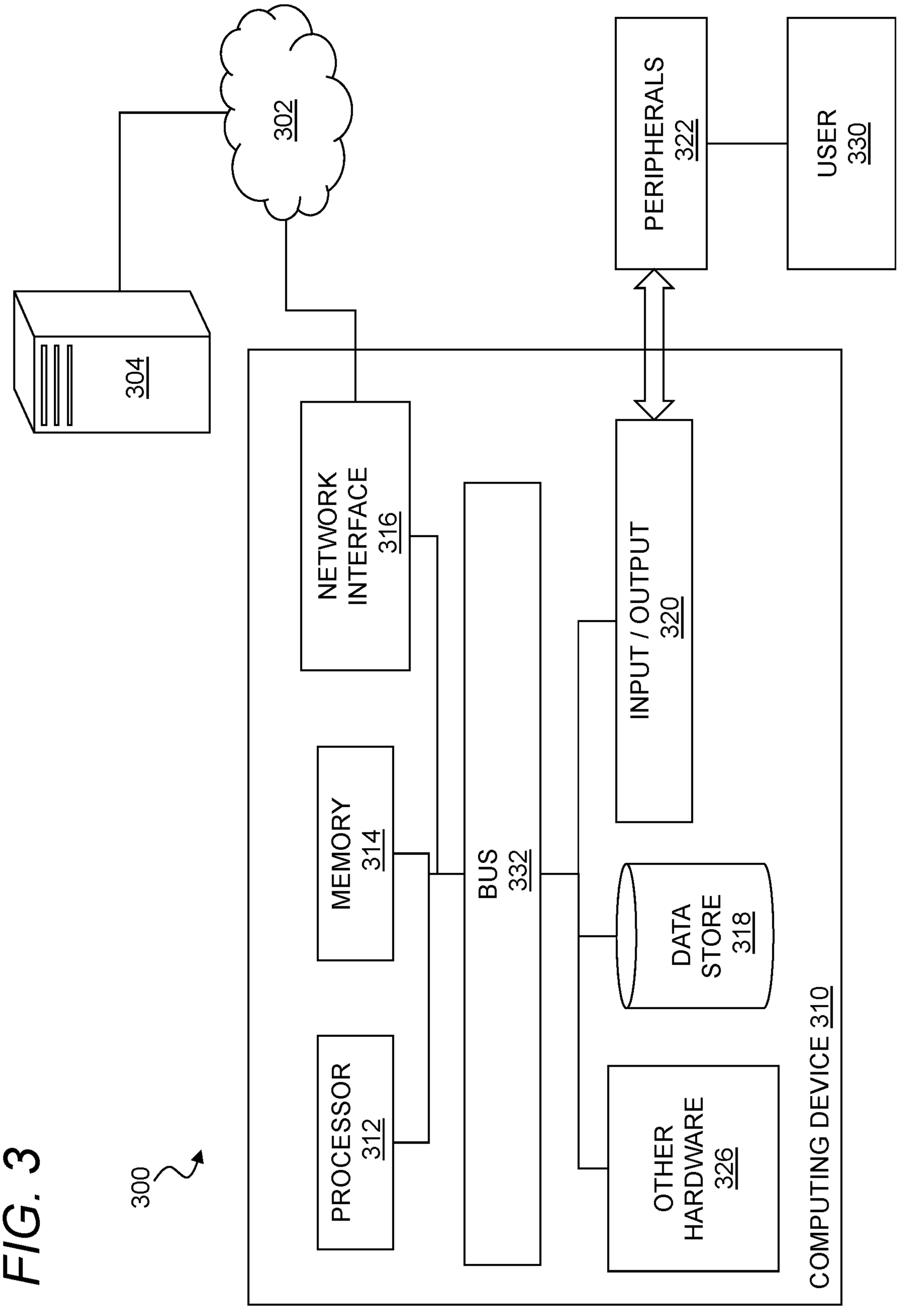

300
302
304
PERIPHERALS
322
USER
330
NETWORK
INTERFACE
316
INPUT / OUTPUT
320
MEMORY
314
PROCESSOR
312
BUS
332
DATA
STORE
318
OTHER
HARDWARE
326
COMPUTING DEVICE 310

FIG. 3

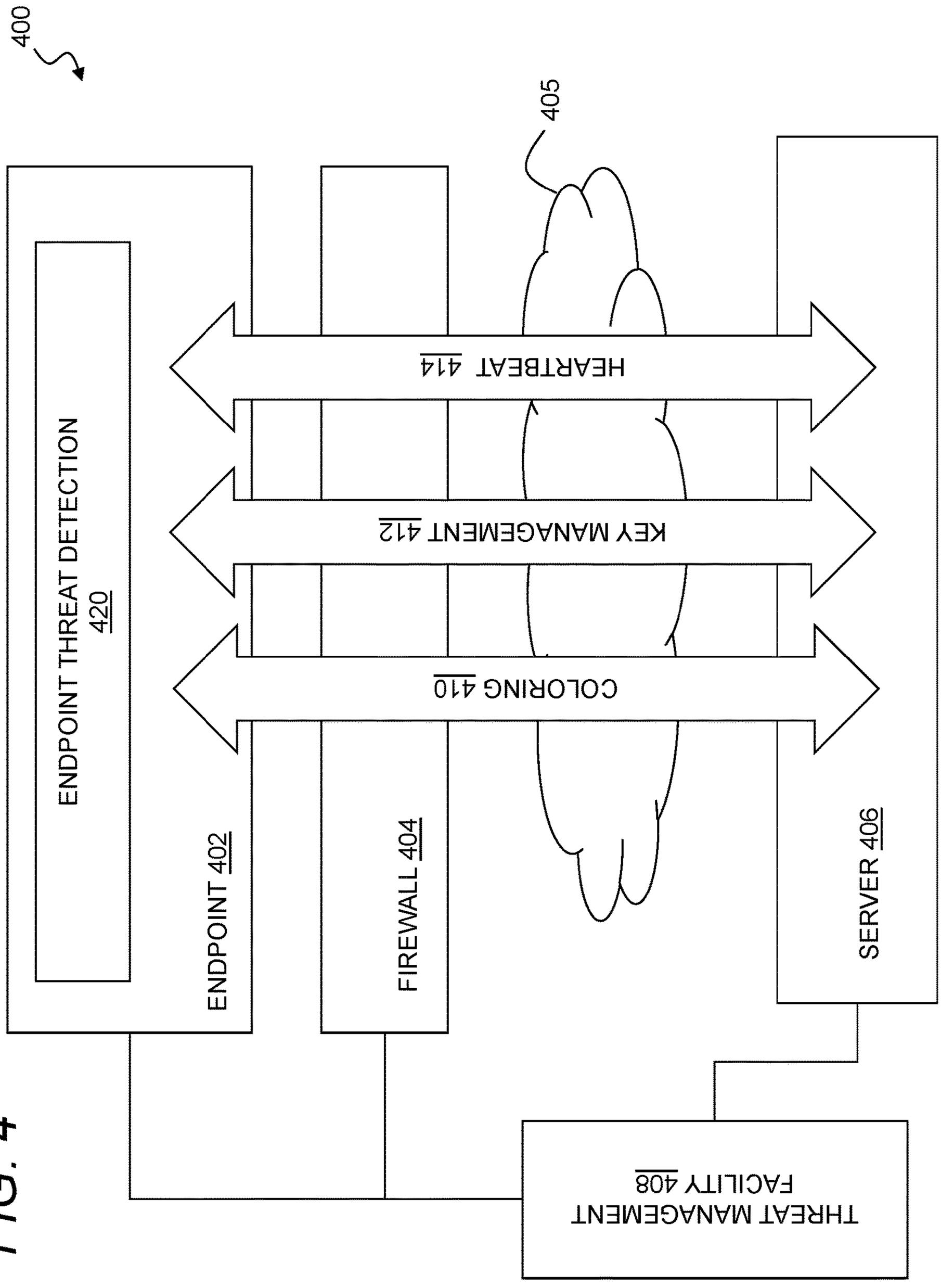

400
405
HEARTBEAT 414
KEY MANAGEMENT 412
COLORING 410
ENDPOINT THREAT DETECTION 420
ENDPOINT 402
FIREWALL 404
SERVER 406
THREAT MANAGEMENT FACILITY 408

FIG. 4

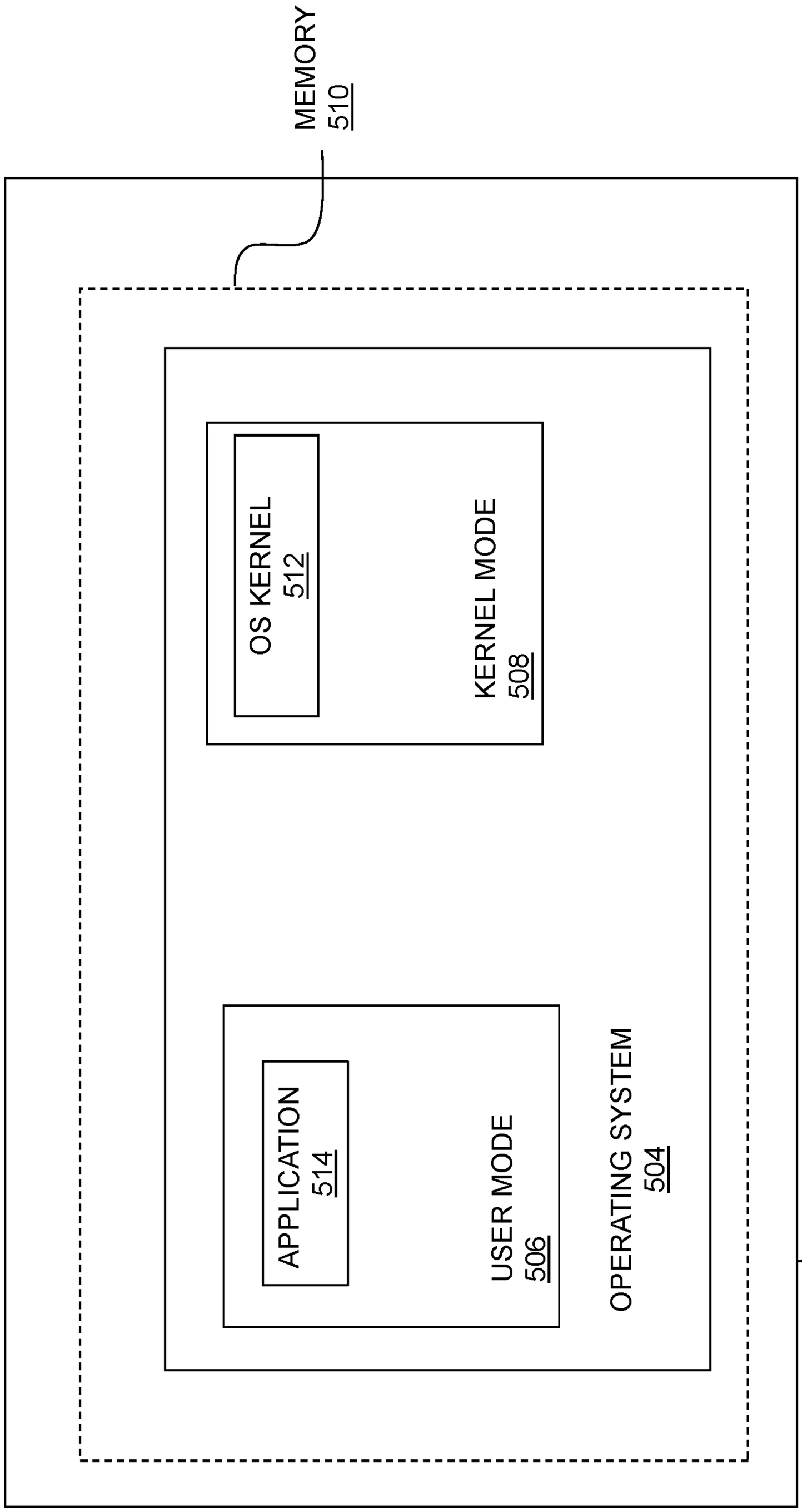

MEMORY
510
OS KERNEL
512
KERNEL MODE
508
APPLICATION
514
USER MODE
506
OPERATING SYSTEM
504
502
500

FIG. 5

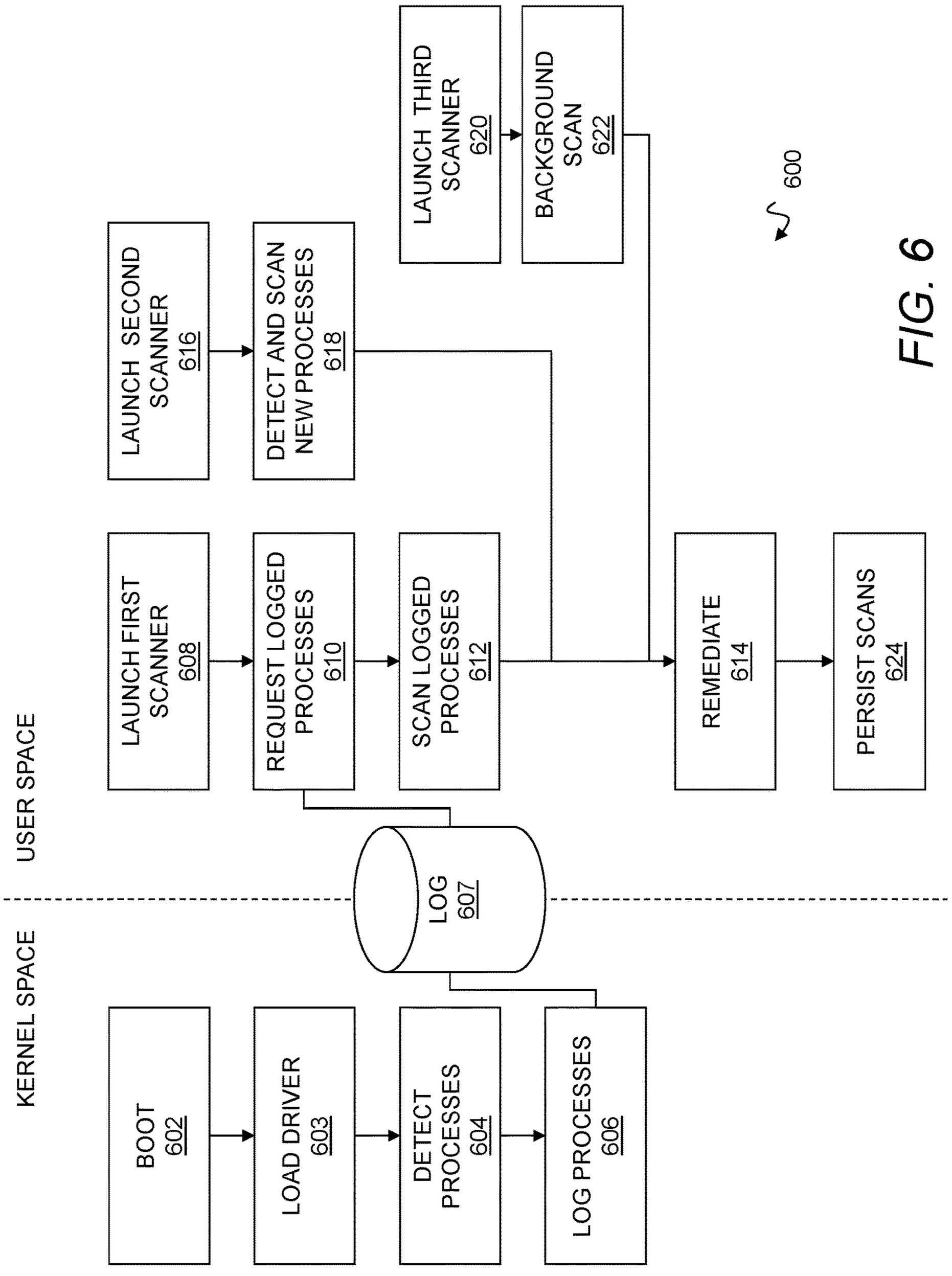

600
KERNEL SPACE
USER SPACE
BOOT 602
LOAD DRIVER 603
DETECT PROCESSES 604
LOG PROCESSES 606
LOG 607
LAUNCH FIRST SCANNER 608
REQUEST LOGGED PROCESSES 610
SCAN LOGGED PROCESSES 612
REMEDIATE 614
LAUNCH SECOND SCANNER 616
DETECT AND SCAN NEW PROCESSES 618
LAUNCH THIRD SCANNER 620
BACKGROUND SCAN 622
PERSIST SCANS 624

FIG. 6

# EARLY BOOT DRIVER FOR START-UP DETECTION OF MALICIOUS CODE

## CROSS-REFERENCE TO RELATED APPLICATIONS

This application is related to the following commonly-owned U.S. patent application Ser. No. 16/437,999 filed on even date herewith and incorporated herein by reference in its entirety: entitled "Early Boot Driver for Start-Up Detection of Malicious Code."

## FIELD

The present disclosure generally relates to a threat management system and threat management techniques, and more particularly to malware mitigation based on detection of malicious processes executing before a user mode scanning tool is available.

## BACKGROUND

Some exploits can evade detection on a compute instance by launching processes early in a boot, e.g., before a corresponding malware scanning tool can begin executing in the user mode. There remains a need for improved detection techniques that can detect and mitigate this type of exploit.

## SUMMARY

A security driver loads early in the boot process for a compute instance and detects processes that are subsequently launched. The detected processes can be recorded, and then scanned with any suitable malware scanning tool(s) once a user mode is available on the compute instance. After the operating system is installed and a user mode is available, other scanning tools may also be deployed (e.g., in the user mode) to augment security of the compute instance.

In one aspect, a computer program product disclosed herein includes computer executable code embodied in a non-transitory computer readable medium that, when executing on a computing device, performs the steps of loading a driver in a kernel mode of an operating system during a boot of the operating system on a compute instance before a user mode of the operating system is available, the driver configured to detect processes starting on the compute instance, storing a list of processes detected by the driver and executing on the compute instance, launching a first scanner in the user mode, the first scanner configured to asynchronously perform a first scan for malware in each process identified in the list of processes from the driver by scanning at least an executable and an executable file path associated with each process, launching a second scanner in the user mode, the second scanner configured to detect one or more other processes started after the first scanner is launched, to synchronously perform a second scan for malware in each of the one or more other processes, and to prevent an execution of each of the one or more other processes until a corresponding scan has been completed, and remediating malicious code identified in at least one of the first scan and the second scan.

The second scanner may begin to scan after the first scanner completes a scan of all of the processes identified in the list of processes. The first scan may include a scan of at least one of an executable associated with one of the processes, files in an executable path associated with one of the processes, or a dynamic linked library loaded by one of the processes. Loading the driver may include registering the driver as a kernel-mode driver for execution at an early stage in a boot process. The driver may record a time stamp indicating a start time for each process identified in the list of processes. The computer program product may further include computer executable code that performs the step of launching a third scanner configured to perform a third scan including a background scan of an entire disk associated with the compute instance. The computer program product may further include computer executable code that performs the step of persisting at least one of the first scan, the second scan, and the third scan with a heartbeat to a threat management facility.

In another aspect, a method disclosed herein includes loading a driver during a boot of an operating system on a compute instance, the driver loaded before a user mode of the operating system is available and the driver configured to store a list of processes executing on the compute instance; launching a first scanner in the user mode, the first scanner configured to perform a first scan for malware in each process identified in the list of processes when the first scanner launches; and launching a second scanner in the user mode, the second scanner configured to detect one or more other processes started after the first scanner is launched, to perform a second scan for malware in each of the one or more other processes, and to prevent an execution of each of the one or more other processes until a corresponding scan has been completed.

The second scanner may synchronously scan the one or more other processes in an order that the one or more other processes launched. The second scanner may begin to scan after the first scanner completes a scan of all processes identified in the list of processes. The first scanner may asynchronously scan processes identified in the list of processes. The first scan may include a scan of at least one of an executable associated with a process in the list of processes, files in an executable path associated with a process in the list of processes, or a dynamic linked library loaded by a process in the list of processes. Loading the driver may include loading the driver early in the boot of the operating system. Loading the driver may include registering the driver as a kernel-mode driver for execution at an early stage in a boot process. The driver may include a certificate for use by a boot time detection driver of the operating system of the compute instance. The driver may record a time stamp indicating a start time for each process identified in the list of processes. The method may further include persisting the first scan with a heartbeat to a threat management facility. The method may further include launching a third scanner, where the third scanner is configured to perform a background scan of an entire disk associated with the compute instance. The method may further include persisting the background scan with a heartbeat to a threat management facility.

In another aspect, a system disclosed herein includes a compute instance, a driver loaded into and executing in a kernel mode of an operating system for the compute instance before a user mode of the operating system is available, the driver configured to record a list of processes executing on the compute instance by recording processes started on the compute instance after the driver is loaded, a first scanner executing in the user mode of the operating system, the first scanner configured to perform a first scan for malware in each process identified in the list of processes at a time that the first scanner launches, and a second scanner executing in the user mode of the operating system, the second scanner configured to detect one or more other processes started after

the second scanner is launched, to perform a second scan for malware in each of the one or more other processes, and to prevent an execution of each of the one or more other processes until a corresponding scan has been completed.

A security driver loads early in the boot process for a compute instance and detects processes that are subsequently launched and/or terminated. The detected processes can be recorded, and then scanned with any suitable malware scanning tool(s) once a user mode is available on the compute instance, including any processes that are terminated before such scanning tools are launched. After the operating system is installed and a user mode is available, other scanning tools may also be deployed (e.g., in the user mode) to augment security of the compute instance.

In one aspect, a computer program product disclosed herein includes computer executable code embodied in a non-transitory computer readable medium that, when executing on a computing device, performs the steps of booting an operating system on a compute instance; during the boot and before a user mode is available, loading a driver in a kernel mode of the operating system, the driver configured to detect and record each process started and stopped on the compute instance as a list of processes; when the user mode is available, launching a scanner in the user mode; requesting, with the scanner, the list of processes detected by the driver, including any processes that have stopped; scanning, with the scanner, one or more files associated with each process in the list of processes; and, if malicious code is identified in the one or more files associated with one of the processes in the list of processes, remediating the one of the processes.

In another aspect, a method disclosed herein includes loading a driver during a boot of an operating system on a compute instance, the driver loaded before a user mode of the operating system is available and the driver configured to record a list of processes including each process started and stopped on the compute instance; launching a scanner in the user mode; requesting, with the scanner, the list of processes recorded by the driver, including any processes that have stopped; and scanning, with the scanner, one or more files associated with each process in the list of processes.

The method may further include launching a second scanner in the user mode, where the second scanner is configured to detect one or more other processes started after the scanner is launched, to perform a second scan for malware in each of the one or more other processes, and to prevent an execution of each of the one or more other processes until a corresponding scan has been completed. The second scanner may begin to scan after the scanner completes a scan of all of the processes identified in the list of processes. The method may further include persisting at least one of the scanner and the second scanner with a heartbeat to a threat management facility. The method may further include launching a third scanner, the third scanner configured to perform a background scan of an entire disk associated with the compute instance. The scanner may asynchronously scan the processes identified in the list of processes. The scanner may scan at least one of an executable associated with a process in the list of processes, files in an executable path associated with a process in the list of processes, or a dynamic linked library loaded by a process in the list of processes. Loading the driver may include loading the driver early in the boot of the operating system. Loading the driver may include registering the driver as a kernel-mode driver for execution at an early stage in a boot process. The driver may include a certificate for use by a boot time detection driver of the operating system of the compute instance. The driver may record a time stamp in the list of processes indicating a start time for each process identified in the list of processes. The method may further include, if malicious code is identified in the one or more files associated with one of the processes in the list of processes that is still executing, remediating the one of the processes. Remediating the one of the processes may include quarantining the one of the processes. Remediating the one of the processes may include executing the one of the processes in a sandbox. Remediating the one of the processes may include restricting access by the one of the processes to one or more resources of the compute instance. The method may further include, if malicious code is identified in the one or more files associated with one of the processes in the list of processes, remediating the one or more files associated with the one of the processes. Remediating the one or more files may include executing a malware detection tool or a malware removal tool for the compute instance. Remediating the one or more files may include remediating the compute instance.

In another aspect, a system disclosed herein includes a compute instance, a driver loaded into and executing in a kernel mode of an operating system for the compute instance before a user mode of the operating system is available, the driver configured to record a list of processes including each process started and stopped on the compute instance after the driver is loaded, a scanner executing in the user mode of the operating system, the scanner configured to perform a scan for malware in each process identified in the list of processes at a time that the scanner launches, and a local security agent executing on the compute instance and configured to remediate malicious code identified by the scanner.

## BRIEF DESCRIPTION OF THE DRAWINGS

The foregoing and other objects, features and advantages of the devices, systems, and methods described herein will be apparent from the following description of particular embodiments thereof, as illustrated in the accompanying drawings. The drawings are not necessarily to scale, emphasis instead being placed upon illustrating the principles of the devices, systems, and methods described herein.

FIG. **1** depicts a block diagram of a threat management system.

FIG. **2** depicts a block diagram of a threat management system.

FIG. **3** illustrates a system for forensic analysis for computer processes.

FIG. **4** illustrates a threat management system.

FIG. **5** shows a block diagram for a computing device.

FIG. **6** shows a method for malware detection.

## DESCRIPTION

Embodiments will now be described with reference to the accompanying figures. The foregoing may, however, be embodied in many different forms and should not be construed as limited to the illustrated embodiments set forth herein.

All documents mentioned herein are hereby incorporated by reference in their entirety. References to items in the singular should be understood to include items in the plural, and vice versa, unless explicitly stated otherwise or clear from the text. Grammatical conjunctions are intended to express any and all disjunctive and conjunctive combina-

tions of conjoined clauses, sentences, words, and the like, unless otherwise stated or clear from the context. Thus, the term "or" should generally be understood to mean "and/or" and so forth.

Recitation of ranges of values herein are not intended to be limiting, referring instead individually to any and all values falling within the range, unless otherwise indicated herein, and each separate value within such a range is incorporated into the specification as if it were individually recited herein. The words "about," "approximately" or the like, when accompanying a numerical value, are to be construed as indicating a deviation as would be appreciated by one of ordinary skill in the art to operate satisfactorily for an intended purpose. Similarly, words of approximation such as "approximately" or "substantially" when used in reference to physical characteristics, should be understood to contemplate a range of deviations that would be appreciated by one of ordinary skill in the art to operate satisfactorily for a corresponding use, function, purpose, or the like. Ranges of values and/or numeric values are provided herein as examples only, and do not constitute a limitation on the scope of the described embodiments. Where ranges of values are provided, they are also intended to include each value within the range as if set forth individually, unless expressly stated to the contrary. The use of any and all examples, or exemplary language ("e.g.," "such as," or the like) provided herein, is intended merely to better illuminate the embodiments and does not pose a limitation on the scope of the embodiments. No language in the specification should be construed as indicating any unclaimed element as essential to the practice of the embodiments.

In the following description, it is understood that terms such as "first," "second," "top," "bottom," "up," "down," and the like, are words of convenience and are not to be construed as limiting terms unless specifically stated to the contrary.

FIG. **1** depicts a block diagram of a threat management system **101** providing protection against a plurality of threats, such as malware, viruses, spyware, cryptoware, adware, Trojans, spam, intrusion, policy abuse, improper configuration, vulnerabilities, improper access, uncontrolled access, and more. A threat management facility **100** may communicate with, coordinate, and control operation of security functionality at different control points, layers, and levels within the system **101**. A number of capabilities may be provided by a threat management facility **100**, with an overall goal to intelligently use the breadth and depth of information that is available about the operation and activity of compute instances and networks as well as a variety of available controls. Another overall goal is to provide protection needed by an organization that is dynamic and able to adapt to changes in compute instances and new threats. In embodiments, the threat management facility **100** may provide protection from a variety of threats to a variety of compute instances in a variety of locations and network configurations.

Just as one example, users of the threat management facility **100** may define and enforce policies that control access to and use of compute instances, networks, and data. Administrators may update policies such as by designating authorized users and conditions for use and access. The threat management facility **100** may update and enforce those policies at various levels of control that are available, such as by directing compute instances to control the network traffic that is allowed to traverse firewalls and wireless access points, applications, and data available from servers, applications, and data permitted to be accessed by endpoints, and network resources and data permitted to be run and used by endpoints. The threat management facility **100** may provide many different services, and policy management may be offered as one of the services.

Turning to a description of certain capabilities and components of the threat management system **101**, an exemplary enterprise facility **102** may be or may include any networked computer-based infrastructure. For example, the enterprise facility **102** may be corporate, commercial, organizational, educational, governmental, or the like. As home networks get more complicated, and include more compute instances at home and in the cloud, an enterprise facility **102** may also or instead include a personal network such as a home or a group of homes. The enterprise facility's **102** computer network may be distributed amongst a plurality of physical premises such as buildings on a campus, and located in one or in a plurality of geographical locations. The configuration of the enterprise facility as shown is merely exemplary, and it will be understood that there may be any number of compute instances, less or more of each type of compute instances, and other types of compute instances. As shown, the exemplary enterprise facility includes a firewall **10**, a wireless access point **11**, an endpoint **12**, a server **14**, a mobile device **16**, an appliance or IOT device **18**, a cloud computing instance **19**, and a server **20**. Again, the compute instances **10-20** depicted are exemplary, and there may be any number or types of compute instances **10-20** in a given enterprise facility. For example, in addition to the elements depicted in the enterprise facility **102**, there may be one or more gateways, bridges, wired networks, wireless networks, virtual private networks, other compute instances, and so on.

The threat management facility **100** may include certain facilities, such as a policy management facility **112**, security management facility **122**, update facility **120**, definitions facility **114**, network access rules facility **124**, remedial action facility **128**, detection techniques facility **130**, application protection facility **150**, asset classification facility **160**, entity model facility **162**, event collection facility **164**, event logging facility **166**, analytics facility **168**, dynamic policies facility **170**, identity management facility **172**, and marketplace management facility **174**, as well as other facilities. For example, there may be a testing facility, a threat research facility, and other facilities. It should be understood that the threat management facility **100** may be implemented in whole or in part on a number of different compute instances, with some parts of the threat management facility on different compute instances in different locations. For example, some or all of one or more of the various facilities **100**, **112-174** may be provided as part of a security agent S that is included in software running on a compute instance **10-26** within the enterprise facility. Some or all of one or more of the facilities **100**, **112-174** may be provided on the same physical hardware or logical resource as a gateway, such as a firewall **10**, or wireless access point **11**. Some or all of one or more of the facilities may be provided on one or more cloud servers that are operated by the enterprise or by a security service provider, such as the cloud computing instance **109**.

In embodiments, a marketplace provider **199** may make available one or more additional facilities to the enterprise facility **102** via the threat management facility **100**. The marketplace provider may communicate with the threat management facility **100** via the marketplace interface facility **174** to provide additional functionality or capabilities to the threat management facility **100** and compute instances **10-26**. A marketplace provider **199** may be selected from a number of providers in a marketplace of providers that are

available for integration or collaboration via the marketplace interface facility **174**. A given marketplace provider **199** may use the marketplace interface facility **174** even if not engaged or enabled from or in a marketplace. As non-limiting examples, the marketplace provider **199** may be a third-party information provider, such as a physical security event provider; the marketplace provider **199** may be a system provider, such as a human resources system provider or a fraud detection system provider; the marketplace provider **199** may be a specialized analytics provider; and so on. The marketplace provider **199**, with appropriate permissions and authorization, may receive and send events, observations, inferences, controls, convictions, policy violations, or other information to the threat management facility. For example, the marketplace provider **199** may subscribe to and receive certain events, and in response, based on the received events and other events available to the marketplace provider **199**, send inferences to the marketplace interface, and in turn to the analytics facility **168**, which in turn may be used by the security management facility **122**.

The identity provider **158** may be any remote identity management system or the like configured to communicate with an identity management facility **172**, e.g., to confirm identity of a user as well as provide or receive other information about users that may be useful to protect against threats. In general, the identity provider may be any system or entity that creates, maintains, and manages identity information for principals while providing authentication services to relying party applications, e.g., within a federation or distributed network. The identity provider may, for example, offer user authentication as a service, where other applications, such as web applications, outsource the user authentication step to a trusted identity provider.

In embodiments, the identity provider **158** may provide user identity information, such as multi-factor authentication, to a SaaS application. Centralized identity providers such as Microsoft Azure, may be used by an enterprise facility instead of maintaining separate identity information for each application or group of applications, and as a centralized point for integrating multifactor authentication. In embodiments, the identity management facility **172** may communicate hygiene, or security risk information, to the identity provider **158**. The identity management facility **172** may determine a risk score for a user based on the events, observations, and inferences about that user and the compute instances associated with the user. If a user is perceived as risky, the identity management facility **172** can inform the identity provider **158**, and the identity provider **158** may take steps to address the potential risk, such as to confirm the identity of the user, confirm that the user has approved the SaaS application access, remediate the user's system, or such other steps as may be useful.

In embodiments, threat protection provided by the threat management facility **100** may extend beyond the network boundaries of the enterprise facility **102** to include clients (or client facilities) such as an endpoint **22** outside the enterprise facility **102**, a mobile device **26**, a cloud computing instance **109**, or any other devices, services or the like that use network connectivity not directly associated with or controlled by the enterprise facility **102**, such as a mobile network, a public cloud network, or a wireless network at a hotel or coffee shop. While threats may come from a variety of sources, such as from network threats, physical proximity threats, secondary location threats, the compute instances **10-26** may be protected from threats even when a compute instance **10-26** is not connected to the enterprise facility **102** network, such as when compute instances **22**, **26** use a network that is outside of the enterprise facility **102** and separated from the enterprise facility **102**, e.g., by a gateway, a public network, and so forth.

In some implementations, compute instances **10-26** may communicate with cloud applications, such as a SaaS application **156**. The SaaS application **156** may be an application that is used by but not operated by the enterprise facility **102**. Exemplary commercially available SaaS applications **156** include Salesforce, Amazon Web Services (AWS) applications, Google Apps applications, Microsoft Office 365 applications, and so on. A given SaaS application **156** may communicate with an identity provider **158** to verify user identity consistent with the requirements of the enterprise facility **102**. The compute instances **10-26** may communicate with an unprotected server (not shown) such as a web site or a third-party application through an internetwork **154** such as the Internet or any other public network, private network, or combination of these.

In embodiments, aspects of the threat management facility **100** may be provided as a stand-alone solution. In other embodiments, aspects of the threat management facility **100** may be integrated into a third-party product. An application programming interface (e.g., a source code interface) may be provided such that aspects of the threat management facility **100** may be integrated into or used by or with other applications. For instance, the threat management facility **100** may be stand-alone in that it provides direct threat protection to an enterprise or computer resource, where protection is subscribed to directly **100**. Alternatively, the threat management facility may offer protection indirectly, through a third-party product, where an enterprise may subscribe to services through the third-party product, and threat protection to the enterprise may be provided by the threat management facility **100** through the third-party product.

The security management facility **122** may provide protection from a variety of threats by providing, as non-limiting examples, endpoint security and control, email security and control, web security and control, reputation-based filtering, machine learning classification, control of unauthorized users, control of guest and non-compliant computers, and more.

The security management facility **122** may provide malicious code protection to a compute instance. The security management facility **122** may include functionality to scan applications, files, and data for malicious code, remove or quarantine applications and files, prevent certain actions, perform remedial actions, as well as other security measures. Scanning may use any of a variety of techniques, including without limitation signatures, identities, classifiers, and other suitable scanning techniques. In embodiments, the scanning may include scanning some or all files on a periodic basis, scanning an application when the application is executed, scanning data transmitted to or from a device, scanning in response to predetermined actions or combinations of actions, and so forth. The scanning of applications, files, and data may be performed to detect known or unknown malicious code or unwanted applications. Aspects of the malicious code protection may be provided, for example, in the security agent of an endpoint **12**, in a wireless access point **11** or firewall **10**, as part of application protection **150** provided by the cloud, and so on.

In an embodiment, the security management facility **122** may provide for email security and control, for example to target spam, viruses, spyware and phishing, to control email content, and the like. Email security and control may protect against inbound and outbound threats, protect email infra-

structure, prevent data leakage, provide spam filtering, and more. Aspects of the email security and control may be provided, for example, in the security agent of an endpoint **12**, in a wireless access point **11** or firewall **10**, as part of application protection **150** provided by the cloud, and so on.

In an embodiment, security management facility **122** may provide for web security and control, for example, to detect or block viruses, spyware, malware, unwanted applications, help control web browsing, and the like, which may provide comprehensive web access control enabling safe, productive web browsing. Web security and control may provide Internet use policies, reporting on suspect compute instances, security and content filtering, active monitoring of network traffic, URI filtering, and the like. Aspects of the web security and control may be provided, for example, in the security agent of an endpoint **12**, in a wireless access point **11** or firewall **10**, as part of application protection **150** provided by the cloud, and so on.

In an embodiment, the security management facility **122** may provide for network access control, which generally controls access to and use of network connections. Network control may stop unauthorized, guest, or non-compliant systems from accessing networks, and may control network traffic that is not otherwise controlled at the client level. In addition, network access control may control access to virtual private networks (VPN), where VPNs may, for example, include communications networks tunneled through other networks and establishing logical connections acting as virtual networks. In embodiments, a VPN may be treated in the same manner as a physical network. Aspects of network access control may be provided, for example, in the security agent of an endpoint **12**, in a wireless access point **11** or firewall **10**, as part of application protection **150** provided by the cloud, e.g., from the threat management facility **100** or other network resource(s).

In an embodiment, the security management facility **122** may provide for host intrusion prevention through behavioral monitoring and/or runtime monitoring, which may guard against unknown threats by analyzing application behavior before or as an application runs. This may include monitoring code behavior, application programming interface calls made to libraries or to the operating system, or otherwise monitoring application activities. Monitored activities may include, for example, reading and writing to memory, reading and writing to disk, network communication, process interaction, and so on. Behavior and runtime monitoring may intervene if code is deemed to be acting in a manner that is suspicious or malicious. Aspects of behavior and runtime monitoring may be provided, for example, in the security agent of an endpoint **12**, in a wireless access point **11** or firewall **10**, as part of application protection **150** provided by the cloud, and so on.

In an embodiment, the security management facility **122** may provide for reputation filtering, which may target or identify sources of known malware. For instance, reputation filtering may include lists of URIs of known sources of malware or known suspicious IP addresses, code authors, code signers, or domains, that when detected may invoke an action by the threat management facility **100**. Based on reputation, potential threat sources may be blocked, quarantined, restricted, monitored, or some combination of these, before an exchange of data can be made. Aspects of reputation filtering may be provided, for example, in the security agent of an endpoint **12**, in a wireless access point **11** or firewall **10**, as part of application protection **150** provided by the cloud, and so on. In embodiments, some reputation information may be stored on a compute instance **10-26**, and other reputation data available through cloud lookups to an application protection lookup database, such as may be provided by application protection **150**.

In embodiments, information may be sent from the enterprise facility **102** to a third party, such as a security vendor, or the like, which may lead to improved performance of the threat management facility **100**. In general, feedback may be useful for any aspect of threat detection. For example, the types, times, and number of virus interactions that an enterprise facility **102** experiences may provide useful information for the preventions of future virus threats. Feedback may also be associated with behaviors of individuals within the enterprise, such as being associated with most common violations of policy, network access, unauthorized application loading, unauthorized external device use, and the like. In embodiments, feedback may enable the evaluation or profiling of client actions that are violations of policy that may provide a predictive model for the improvement of enterprise policies.

An update management facility **120** may provide control over when updates are performed. The updates may be automatically transmitted, manually transmitted, or some combination of these. Updates may include software, definitions, reputations or other code or data that may be useful to the various facilities. For example, the update facility **120** may manage receiving updates from a provider, distribution of updates to enterprise facility **102** networks and compute instances, or the like. In embodiments, updates may be provided to the enterprise facility's **102** network, where one or more compute instances on the enterprise facility's **102** network may distribute updates to other compute instances.

The threat management facility **100** may include a policy management facility **112** that manages rules or policies for the enterprise facility **102**. Exemplary rules include access permissions associated with networks, applications, compute instances, users, content, data, and the like. The policy management facility **112** may use a database, a text file, other data store, or a combination to store policies. In an embodiment, a policy database may include a block list, a black list, an allowed list, a white list, and more. As a few non-limiting examples, policies may include a list of enterprise facility **102** external network locations/applications that may or may not be accessed by compute instances, a list of types/classifications of network locations or applications that may or may not be accessed by compute instances, and contextual rules to evaluate whether the lists apply. For example, there may be a rule that does not permit access to sporting websites. When a website is requested by the client facility, a security management facility **122** may access the rules within a policy facility to determine if the requested access is related to a sporting website.

The policy management facility **112** may include access rules and policies that are distributed to maintain control of access by the compute instances **10-26** to network resources. Exemplary policies may be defined for an enterprise facility, application type, subset of application capabilities, organization hierarchy, compute instance type, user type, network location, time of day, connection type, or any other suitable definition. Policies may be maintained through the threat management facility **100**, in association with a third party, or the like. For example, a policy may restrict instant messaging (IM) activity by limiting such activity to support personnel when communicating with customers. More generally, this may allow communication for departments as necessary or helpful for department functions, but may otherwise preserve network bandwidth for other activities by restricting the use of IM to personnel that need access for a

specific purpose. In an embodiment, the policy management facility **112** may be a stand-alone application, may be part of the network server facility **142**, may be part of the enterprise facility **102** network, may be part of the client facility, or any suitable combination of these.

The policy management facility **112** may include dynamic policies that use contextual or other information to make security decisions. As described herein, the dynamic policies facility **170** may generate policies dynamically based on observations and inferences made by the analytics facility. The dynamic policies generated by the dynamic policy facility **170** may be provided by the policy management facility **112** to the security management facility **122** for enforcement.

In embodiments, the threat management facility **100** may provide configuration management as an aspect of the policy management facility **112**, the security management facility **122**, or some combination. Configuration management may define acceptable or required configurations for the compute instances **10-26**, applications, operating systems, hardware, or other assets, and manage changes to these configurations. Assessment of a configuration may be made against standard configuration policies, detection of configuration changes, remediation of improper configurations, application of new configurations, and so on. An enterprise facility may have a set of standard configuration rules and policies for particular compute instances which may represent a desired state of the compute instance. For example, on a given compute instance **12**, **14**, **18**, a version of a client firewall may be required to be running and installed. If the required version is installed but in a disabled state, the policy violation may prevent access to data or network resources. A remediation may be to enable the firewall. In another example, a configuration policy may disallow the use of USB disks, and policy management **112** may require a configuration that turns off USB drive access via a registry key of a compute instance. Aspects of configuration management may be provided, for example, in the security agent of an endpoint **12**, in a wireless access point **11** or firewall **10**, as part of application protection **150** provided by the cloud, or any combination of these.

In embodiments, the threat management facility **100** may also provide for the isolation or removal of certain applications that are not desired or may interfere with the operation of a compute instance **10-26** or the threat management facility **100**, even if such application is not malware per se. The operation of such products may be considered a configuration violation. The removal of such products may be initiated automatically whenever such products are detected, or access to data and network resources may be restricted when they are installed and running. In the case where such applications are services which are provided indirectly through a third-party product, the applicable application or processes may be suspended until action is taken to remove or disable the third-party product.

The policy management facility **112** may also require update management (e.g., as provided by the update facility **120**). Update management for the security facility **122** and policy management facility **112** may be provided directly by the threat management facility **100**, or, for example, by a hosted system. In embodiments, the threat management facility **100** may also provide for patch management, where a patch may be an update to an operating system, an application, a system tool, or the like, where one of the reasons for the patch is to reduce vulnerability to threats.

In embodiments, the security facility **122** and policy management facility **112** may push information to the enterprise facility **102** network and/or the compute instances **10-26**, the enterprise facility **102** network and/or compute instances **10-26** may pull information from the security facility **122** and policy management facility **112**, or there may be a combination of pushing and pulling of information. For example, the enterprise facility **102** network and/or compute instances **10-26** may pull update information from the security facility **122** and policy management facility **112** via the update facility **120**, an update request may be based on a time period, by a certain time, by a date, on demand, or the like. In another example, the security facility **122** and policy management facility **112** may push the information to the enterprise facility's **102** network and/or compute instances **10-26** by providing notification that there are updates available for download and/or transmitting the information. In an embodiment, the policy management facility **112** and the security facility **122** may work in concert with the update management facility **120** to provide information to the enterprise facility's **102** network and/or compute instances **10-26**. In various embodiments, policy updates, security updates and other updates may be provided by the same or different modules, which may be the same or separate from a security agent running on one of the compute instances **10-26**.

As threats are identified and characterized, the definition facility **114** of the threat management facility **100** may manage definitions used to detect and remediate threats. For example, identity definitions may be used for scanning files, applications, data streams, etc. for the determination of malicious code. Identity definitions may include instructions and data that can be parsed and acted upon for recognizing features of known or potentially malicious code. Definitions also may include, for example, code or data to be used in a classifier, such as a neural network or other classifier that may be trained using machine learning. Updated code or data may be used by the classifier to classify threats. In embodiments, the threat management facility **100** and the compute instances **10-26** may be provided with new definitions periodically to include most recent threats. Updating of definitions may be managed by the update facility **120**, and may be performed upon request from one of the compute instances **10-26**, upon a push, or some combination. Updates may be performed upon a time period, on demand from a device **10-26**, upon determination of an important new definition or a number of definitions, and so on.

A threat research facility (not shown) may provide a continuously ongoing effort to maintain the threat protection capabilities of the threat management facility **100** in light of continuous generation of new or evolved forms of malware. Threat research may be provided by researchers and analysts working on known threats, in the form of policies, definitions, remedial actions, and so on.

The security management facility **122** may scan an outgoing file and verify that the outgoing file is permitted to be transmitted according to policies. By checking outgoing files, the security management facility **122** may be able discover threats that were not detected on one of the compute instances **10-26**, or policy violation, such transmittal of information that should not be communicated unencrypted.

The threat management facility **100** may control access to the enterprise facility **102** networks. A network access facility **124** may restrict access to certain applications, networks, files, printers, servers, databases, and so on. In addition, the network access facility **124** may restrict user access under certain conditions, such as the user's location, usage history, need to know, job position, connection type, time of day, method of authentication, client-system con-

figuration, or the like. Network access policies may be provided by the policy management facility **112**, and may be developed by the enterprise facility **102**, or pre-packaged by a supplier. Network access facility **124** may determine if a given compute instance **10-22** should be granted access to a requested network location, e.g., inside or outside of the enterprise facility **102**. Network access facility **124** may determine if a compute instance **22**, **26** such as a device outside the enterprise facility **102** may access the enterprise facility **102**. For example, in some cases, the policies may require that when certain policy violations are detected, certain network access is denied. The network access facility **124** may communicate remedial actions that are necessary or helpful to bring a device back into compliance with policy as described below with respect to the remedial action facility **128**. Aspects of the network access facility **124** may be provided, for example, in the security agent of the endpoint **12**, in a wireless access point **11**, in a firewall **10**, as part of application protection **150** provided by the cloud, and so on.

In an embodiment, the network access facility **124** may have access to policies that include one or more of a block list, a black list, an allowed list, a white list, an unacceptable network site database, an acceptable network site database, a network site reputation database, or the like of network access locations that may or may not be accessed by the client facility. Additionally, the network access facility **124** may use rule evaluation to parse network access requests and apply policies. The network access rule facility **124** may have a generic set of policies for all compute instances, such as denying access to certain types of websites, controlling instant messenger accesses, or the like. Rule evaluation may include regular expression rule evaluation, or other rule evaluation method(s) for interpreting the network access request and comparing the interpretation to established rules for network access. Classifiers may be used, such as neural network classifiers or other classifiers that may be trained by machine learning.

The threat management facility **100** may include an asset classification facility **160**. The asset classification facility will discover the assets present in the enterprise facility **102**. A compute instance such as any of the compute instances **10-26** described herein may be characterized as a stack of assets. The one level asset is an item of physical hardware. The compute instance may be, or may be implemented on physical hardware, and may have or may not have a hypervisor, or may be an asset managed by a hypervisor. The compute instance may have an operating system (e.g., Windows, MacOS, Linux, Android, iOS). The compute instance may have one or more layers of containers. The compute instance may have one or more applications, which may be native applications, e.g., for a physical asset or virtual machine, or running in containers within a computing environment on a physical asset or virtual machine, and those applications may link libraries or other code or the like, e.g., for a user interface, cryptography, communications, device drivers, mathematical or analytical functions and so forth. The stack may also interact with data. The stack may also or instead interact with users, and so users may be considered assets.

The threat management facility may include entity models **162**. The entity models **162** may be used, for example, to determine the events that are generated by assets. For example, some operating systems may provide useful information for detecting or identifying events. For examples, operating systems may provide process and usage information that accessed through an API. As another example, it may be possible to instrument certain containers to monitor the activity of applications running on them. As another example, entity models for users may define roles, groups, permitted activities, and other attributes.

The event collection facility **164** may be used to collect events from any of a wide variety of sensors that may provide relevant events from an asset, such as sensors on any of the compute instances **10-26**, the application protection facility **150**, a cloud computing instance **109** and so on. The events that may be collected may be determined by the entity models. There may be a variety of events collected. Events may include, for example, events generated by the enterprise facility **102** or the compute instances **10-26**, such as by monitoring streaming data through a gateway such as firewall **10** and wireless access point **11**, monitoring activity of compute instances, monitoring stored files/data on the compute instances **10-26** such as desktop computers, laptop computers, other mobile computing devices, and cloud computing instances **19**, **109**. Events may range in granularity. An exemplary event may be communication of a specific packet over the network. Another exemplary event may be identification of an application that is communicating over a network.

The event logging facility **166** may be used to store events collected by the event collection facility **164**. The event logging facility **166** may store collected events so that they can be accessed and analyzed by the analytics facility **168**. Some events may be collected locally, and some events may be communicated to an event store in a central location or cloud facility. Events may be logged in any suitable format.

Events collected by the event logging facility **166** may be used by the analytics facility **168** to make inferences and observations about the events. These observations and inferences may be used as part of policies enforced by the security management facility. Observations or inferences about events may also be logged by the event logging facility **166**.

When a threat or other policy violation is detected by the security management facility **122**, the remedial action facility **128** may be used to remediate the threat. Remedial action may take a variety of forms, non-limiting examples including collecting additional data about the threat, terminating or modifying an ongoing process or interaction, sending a warning to a user or administrator, downloading a data file with commands, definitions, instructions, or the like to remediate the threat, requesting additional information from the requesting device, such as the application that initiated the activity of interest, executing a program or application to remediate against a threat or violation, increasing telemetry or recording interactions for subsequent evaluation, (continuing to) block requests to a particular network location or locations, scanning a requesting application or device, quarantine of a requesting application or the device, isolation of the requesting application or the device, deployment of a sandbox, blocking access to resources, e.g., a USB port, or other remedial actions. More generally, the remedial action facility **122** may take any steps or deploy any measures suitable for addressing a detection of a threat, potential threat, policy violation or other event, code, or activity that might compromise security of a computing instance **10-26** or the enterprise facility **102**.

FIG. **2** depicts a block diagram of a threat management system **201** such as any of the threat management systems described herein, and including a cloud enterprise facility **280**. The cloud enterprise facility **280** may include servers **284**, **286**, and a firewall **282**. The servers **284**, **286** on the cloud enterprise facility **280** may run one or more enterprise

applications and make them available to the enterprise facilities **102** compute instances **10-26**. It should be understood that there may be any number of servers **284**, **286** and firewalls **282**, as well as other compute instances in a given cloud enterprise facility **280**. It also should be understood that a given enterprise facility may use both SaaS applications **156** and cloud enterprise facilities **280**, or, for example, a SaaS application **156** may be deployed on a cloud enterprise facility **280**. As such, the configurations in FIG. **1** and FIG. **2** are shown by way of examples and not exclusive alternatives.

Having provided an overall context for threat detection, the description now turns to a brief discussion of an example of a computer system that may be used for any of the entities and facilities described above.

FIG. **3** illustrates a computer system. In general, the computer system **300** may include a computing device **310** connected to a network **302**, e.g., through an external device **304**. The computing device **310** may be or include any type of network endpoint or endpoints as described herein, e.g., with reference to FIG. **1** above. For example, the computing device **310** may include a desktop computer workstation. The computing device **310** may also or instead be any other device that has a processor and communicates over a network **302**, including without limitation a laptop computer, a desktop computer, a personal digital assistant, a tablet, a mobile phone, a television, a set top box, a wearable computer (e.g., watch, jewelry, or clothing), a home device (e.g., a thermostat, humidistat, appliance or a home appliance controller), just as some examples. The computing device **310** may also or instead include a server, or it may be disposed on a server or within a virtual or physical server farm.

The computing device **310** may be any of the entities in the threat management environment described above with reference to FIG. **1**. For example, the computing device **310** may be a server, a client an enterprise facility, a threat management facility, or any of the other facilities or computing devices described therein. In certain aspects, the computing device **310** may be implemented using hardware (e.g., in a desktop computer), software (e.g., in a virtual machine or the like), or a combination of software and hardware (e.g., with programs executing on the desktop computer), and the computing device **310** may be a standalone device, a device integrated into another entity or device, a platform distributed across multiple entities, or a virtualized device executing in a virtualization environment.

The network **302** may include any network or combination of networks, such as one or more data networks or internetworks suitable for communicating data and control information among participants in the computer system **300**. The network **302** may include public networks such as the Internet, private networks, and telecommunications networks such as the Public Switched Telephone Network or cellular networks using third generation cellular technology (e.g., 3G or IMT-2000), fourth generation cellular technology (e.g., 4G, LTE. MT-Advanced, E-UTRA, etc.) or WiMax-Advanced (IEEE 802.16m)) and/or other technologies, as well as any of a variety of corporate area, metropolitan area, campus or other local area networks or enterprise networks, along with any switches, routers, hubs, gateways, and the like that might be used to carry data among participants in the computer system **300**. The network **302** may also include a combination of data networks, and need not be limited to a strictly public or private network.

The external device **304** may be any computer or other remote resource that connects to the computing device **310** through the network **302**. This may include threat management resources such as any of those contemplated above, gateways or other network devices, remote servers or the like containing content requested by the computing device **310**, a network storage device or resource, a device hosting content, or any other resource or device that might connect to the computing device **310** through the network **302**.

The computing device **310** may include a processor **312**, a memory **314**, a network interface **316**, a data store **318**, and one or more input/output devices **320**. The computing device **310** may further include or be in communication with one or more peripherals **322** and other external input/output devices **224**.

The processor **312** may be any as described herein, and in general may be capable of processing instructions for execution within the computing device **310** or computer system **300**. The processor **312** may include a single-threaded processor, a multi-threaded processor, a multi-core processor, or any other processor, processing circuitry, or combination of the foregoing suitable for processing data and instructions as contemplated herein. The processor **312** may be capable of processing instructions stored in the memory **314** or on the data store **318**.

The memory **314** may store information within the computing device **310** or computer system **300**. The memory **314** may include any volatile or non-volatile memory or other computer-readable medium, including without limitation a Random-Access Memory (RAM), a flash memory, a Read Only Memory (ROM), a Programmable Read-only Memory (PROM), an Erasable PROM (EPROM), registers, and so forth. The memory **314** may store program instructions, program data, executables, and other software and data useful for controlling operation of the computing device **310** and configuring the computing device **310** to perform functions for a user. The memory **314** may include a number of different stages and types for different aspects of operation of the computing device **310**. For example, a processor may include on-board memory and/or cache for faster access to certain data or instructions, and a separate, main memory or the like may be included to expand memory capacity as desired.

The memory **314** may, in general, include a non-volatile computer readable medium containing computer code that, when executed by the computing device **310** creates an execution environment for a computer program in question, e.g., code that constitutes processor firmware, a protocol stack, a database management system, an operating system, or a combination of the foregoing, and/or code that performs some or all of the steps set forth in the various flow charts and other algorithmic descriptions set forth herein. While a single memory **314** is depicted, it will be understood that any number of memories may be usefully incorporated into the computing device **310**. For example, a first memory may provide non-volatile storage such as a disk drive for permanent or long-term storage of files and code even when the computing device **310** is powered down. A second memory such as a random-access memory may provide volatile (but higher speed) memory for storing instructions and data for executing processes. A third memory may be used to improve performance by providing even higher speed memory physically adjacent to the processor **312** for registers, caching and so forth.

The network interface **316** may include any hardware and/or software for connecting the computing device **310** in a communicating relationship with other resources through

the network **302**. This may include connections to resources such as remote resources accessible through the Internet, as well as local resources available using short range communications protocols using, e.g., physical connections (e.g., Ethernet), radio frequency communications (e.g., WiFi), optical communications, (e.g., fiber optics, infrared, or the like), ultrasonic communications, or any combination of these or other media that might be used to carry data between the computing device **310** and other devices. The network interface **316** may, for example, include a router, a modem, a network card, an infrared transceiver, a radio frequency (RF) transceiver, a near field communications interface, a radio-frequency identification (RFID) tag reader, or any other data reading or writing resource or the like.

More generally, the network interface **316** may include any combination of hardware and software suitable for coupling the components of the computing device **310** to other computing or communications resources. By way of example and not limitation, this may include electronics for a wired or wireless Ethernet connection operating according to the IEEE 802.11 standard (or any variation thereof), or any other short or long range wireless networking components or the like. This may also or instead include hardware for short range data communications such as Bluetooth or an infrared transceiver, which may be used to couple to other local devices, or to connect to a local area network or the like that is in turn coupled to a data network **302** such as the Internet. This may also or instead include hardware/software for a WiMax connection or a cellular network connection (using, e.g., CDMA, GSM, LTE, or any other suitable protocol or combination of protocols). The network interface **316** may be included as part of the input/output devices **320** or vice-versa.

The data store **318** may be any internal memory store providing a computer-readable medium such as a disk drive, an optical drive, a magnetic drive, a flash drive, or other device capable of providing mass storage for the computing device **310**. The data store **318** may store computer readable instructions, data structures, program modules, and other data for the computing device **310** or computer system **300** in a non-volatile form for subsequent retrieval and use. The data store **318** may store computer executable code for an operating system, application programs, and other program modules, software objects, libraries, executables, and the like the like. The data store **318** may also store program data, databases, files, media, and so forth.

The input/output interface **320** may support input from and output to other devices that might couple to the computing device **310**. This may, for example, include serial ports (e.g., RS-232 ports), universal serial bus (USB) ports, optical ports, Ethernet ports, telephone ports, audio jacks, component audio/video inputs, HDMI ports, and so forth, any of which might be used to form wired connections to other local devices. This may also or instead include an infrared interface, RF interface, magnetic card reader, or other input/output system for coupling in a communicating relationship with other local devices. It will be understood that, while the network interface **316** for network communications is described separately from the input/output interface **320** for local device communications, these two interfaces may be the same, or may share functionality, such as where a USB port is used to attach to a WiFi accessory or other network interfacing device, or where an Ethernet connection is used to couple to a local network attached storage.

The peripherals **322** may include any device or combination of devices used to provide information to or receive information from the computing device **310**. This may include human input/output (I/O) devices such as a keyboard, a mouse, a mouse pad, a track ball, a joystick, a microphone, a foot pedal, a camera, a touch screen, a scanner, or other device that might be employed by the user **330** to provide input to the computing device **310**. This may also or instead include a display, a speaker, a printer, a projector, a headset or any other audiovisual device for presenting information to a user or otherwise providing machine-usable or human-usable output from the computing device **310**. The peripheral **322** may also or instead include a digital signal processing device, an actuator, or other device to support control of or communication with other devices or components. Other I/O devices suitable for use as a peripheral **322** include haptic devices, three-dimensional rendering systems, augmented-reality displays, magnetic card readers, three-dimensional printers, computer-numerical controlled manufacturing machines and so forth. In one aspect, the peripheral **322** may serve as the network interface **316**, such as with a USB device configured to provide communications via short range (e.g., Bluetooth, WiFi, Infrared, RF, or the like) or long range (e.g., cellular data or WiMax) communications protocols. In another aspect, the peripheral **322** may provide a device to augment operation of the computing device **310**, such as a global positioning system (GPS) device, a security dongle, a projector, or the like. In another aspect, the peripheral may be a storage device such as a flash card, USB drive, or other solid-state device, or an optical drive, a magnetic drive, a disk drive, or other device or combination of devices suitable for bulk storage. More generally, any device or combination of devices suitable for use with the computing device **310** may be used as a peripheral **322** as contemplated herein.

Other hardware **326** may be incorporated into the computing device **310** such as a co-processor, a digital signal processing system, a math co-processor, a graphics engine, a video driver, and so forth. The other hardware **326** may also or instead include expanded input/output ports, extra memory, additional drives (e.g., a DVD drive or other accessory), and so forth.

A bus **332** or combination of busses may serve as an electromechanical platform for interconnecting components of the computing device **310** such as the processor **312**, memory **314**, network interface **316**, other hardware **326**, data store **318**, and input/output interface. As shown in the figure, each of the components of the computing device **310** may be interconnected using a system bus **332** or other communication mechanism for communicating information.

Methods and systems described herein can be realized using the processor **312** of the computer system **300** to execute one or more sequences of instructions contained in the memory **314** to perform predetermined tasks. In embodiments, the computing device **310** may be deployed as a number of parallel processors synchronized to execute code together for improved performance, or the computing device **310** may be realized in a virtualized environment where software on a hypervisor or other virtualization management facility emulates components of the computing device **310** as appropriate to reproduce some or all of the functions of a hardware instantiation of the computing device **310**.

FIG. **4** illustrates a threat management system according to some implementations. In general, the system **400** may include an endpoint **402**, a firewall **404**, a server **406**, and a threat management facility **408** coupled to one another directly or indirectly through a data network **405**, all as generally described above. Each of the entities depicted in FIG. **4** may, for example, be implemented on one or more

computing devices such as the computing device described above. A number of systems may be distributed across these various components to support threat detection, such as a coloring system **410**, a key management system **412**, and a heartbeat system **414** (or otherwise an endpoint health system), each of which may include software components executing on any of the foregoing system components, and each of which may communicate with the threat management facility **408** and an endpoint threat detection agent **420** executing on the endpoint **402** to support improved threat detection and remediation.

The coloring system **410** may be used to label or 'color' software objects for improved tracking and detection of potentially harmful activity. The coloring system **410** may, for example, label files, executables, processes, network communications, data sources, and so forth with any suitable label. A variety of techniques may be used to select static and/or dynamic labels for any of these various software objects, and to manage the mechanics of applying and propagating coloring information as appropriate. For example, a process may inherit a color from an application that launches the process. Similarly, a file may inherit a color from a process when it is created or opened by a process, and/or a process may inherit a color from a file that the process has opened. More generally, any type of labeling, as well as rules for propagating, inheriting, changing, or otherwise manipulating such labels, may be used by the coloring system **410** as contemplated herein.

The key management system **412** may support management of keys for the endpoint **402** in order to selectively permit or prevent access to content on the endpoint **402** on a file-specific basis, a process-specific basis, an application-specific basis, a user-specific basis, or any other suitable basis in order to prevent data leakage, and in order to support more fine-grained and immediate control over access to content on the endpoint **402** when a security compromise is detected. Thus, for example, if a particular process executing on the endpoint is compromised, or potentially compromised or otherwise under suspicion, access by that process may be blocked (e.g., with access to keys revoked) in order to prevent, e.g., data leakage or other malicious activity.

The heartbeat system **414** may be used to provide periodic or aperiodic information from the endpoint **402** or other system components about system health, security, status, and so forth. The heartbeat system **414** or otherwise an endpoint health system may thus in general include a health status report system for the endpoint **402**, such as through the use of a heartbeat system or the like. A heartbeat may be encrypted or plaintext, or some combination of these, and may be communicated unidirectionally (e.g., from the endpoint **408** to the threat management facility **408**) or bidirectionally (e.g., between the endpoint **402** and the server **406**, or any other pair of system components) on any useful schedule.

In general, these various monitoring and management systems may cooperate to provide improved threat detection and response. For example, the coloring system **410** may be used to evaluate when a particular process is potentially opening inappropriate files, and a potential threat may be confirmed based on an interrupted heartbeat from the heartbeat system **414**. The key management system **412** may then be deployed to revoke access by the process to certain resources (e.g., keys or file) so that no further files can be opened, deleted, or otherwise modified. More generally, the cooperation of these systems enables a wide variety of reactive measures that can improve detection and remediation of potential threats to an endpoint.

FIG. **5** shows a block diagram of a computing system. In general, the computing system **500** may include a compute instance **502**, such as a virtual device, physical device, or any of the other endpoints or compute instances described herein, executing an operating system **504**. The compute instance **502** may include a processor executing in a user mode **506** and/or a kernel mode **508**, along with memory **510** which may be partitioned into a corresponding user space and kernel space. The user space provides memory for general use, e.g., for the user mode **508**, while the kernel space provides memory for exclusive use by the kernel mode **508**. This latter memory—the kernel space—is generally protected against access from the user mode **506**. In general, the kernel mode **508** supports the operation and use of a computing system with an operating system kernel **512**, along with any file system drivers, kernel-mode drivers, and a hardware abstraction layer for access to hardware, physical memory, and the like. The user mode **506** provides an environment for users to run applications. While the user mode **506** is non-privileged, and cannot access portions of memory allocated to the kernel mode **508**, the kernel mode **508** will generally have root access permissions to access any memory space or other system resources. This general architecture helps to protect the operating system kernel **512** against accidental or intentional (e.g., malicious) interference by applications **514** executing in the user mode **506**. While the foregoing is representative of the structure of many modern operating systems, it will be understood that the terminology and architecture may vary without departing from the present disclosure, which may be useful in any environment where a progressive boot process or the like builds an operating system or other operating environment before permitting execution of applications, and in particular, malware scanners or the like.

In general, a boot or start-up sequence for the compute instance **502** may initiate a progressive execution of code in segments that generally increase in size and complexity until the full operating system **504** is deployed. For example, the boot may begin with an initial read of a boot record or the like from a bootable memory that includes a partition table identifying identifies a file system. The boot record may also include boot code to process the partition table and identify a bootable partition that includes, e.g., an operating-system-specific boot sector. After the boot, the boot record may then transfer control to the boot sector which may in turn detect and configure hardware as appropriate, and progressively load the operating system kernel **512** (and related items) in the kernel mode **508** of the operating system **504**. The operating system **504** may more generally include, e.g., an operating system kernel **512** that supports core functions of the operating system **504**, an application programming interface to the operating system kernel **512**, a file system, a user interface, device drivers, hardware devices, and so forth. Once the operating system **504** has been launched, a user mode **506** is available for use, e.g., by a human or computer user, in executing an application **514**. This may include any application, combination of applications, processes, or the like suitable for execution within the user mode **506** of a compute instance **502**.

In general, this architecture works well, and provides for a secure, stable operating system platform upon which user applications can be launched. However, one security exposure is a vulnerability to malware that launches early in the boot process, e.g., before the operating system is installed to support programs such as malware scanners or other code that might otherwise detect and respond to malicious activity. Various techniques have been developed to address this

potential exposure, such as the Early Launch Anti-Malware (ELAM) module introduced in Microsoft Corporation's Windows **8** operating system, and the Kernel-Mode Code Signing Policy, introduced in Windows Vista. For example, the Kernel-Mode Code Signing Policy generally protects the operating system by imposing digital signature requirements on items loaded into the kernel mode **508**. Similarly, the ELAM module is a detection mechanism that facilitates loading and execution of registered third-party code early in the boot process, and classifies boot-start drivers for conditional execution during the boot. However, because these and similar modules launch early in the boot process, they can be significantly constrained in terms of computing resources, and they typically operate on very limited information sets such as a single filename, hash, signature, or the like for each new boot item. As such, it can be difficult to balance the computational tax of false positives and negatives with a speedy and error free boot process.

To address these challenges, the techniques described herein employ a relatively simple boot driver that detects and logs each new process as a device boots and loads and operating system, combined with a robust, user-mode malware scanner that is loaded after a boot is completed and applied to scan data associated with the logged processes. In this manner, the full computational and contextual resources of the operating system **504** and user mode **506** for a compute instance are available to analyze (and remediate, as necessary) processes launched early in a boot process.

FIG. **6** shows a method for malware detection. In the method **600**, a security driver loads early in the boot process for a compute instance and detects processes that are subsequently launched. The detected processes can be recorded, and then scanned with any suitable malware scanning tool(s) once a user mode is available on the compute instance, optionally including a scan of processes that have terminated before any user mode scanning tools have launched. After the operating system is installed and a user mode is available, other scanning tools may also be deployed (e.g., in the user mode) to augment security of the compute instance.

As shown in step **602**, the process **600** may begin with a boot of a compute instance. As described above, the start-up sequence may begin with a boot of an operating system on a compute instance, and initiate a progressive execution of code in segments that generally increase in size and complexity until a full operating system is deployed on the compute instance and available for execution of user mode applications and the like.

As shown in step **603**, the method **600** may include loading a driver to detect and record processes launched during startup. In general, this driver is preferably loaded as early as possible in the startup process in order to increase the detection of boot and startup processes executing on the compute instance. For example, loading the driver may include loading a driver, e.g., in a kernel mode of an operating system, during a boot of the operating system on a compute instance before a user mode of the operating system is available, or otherwise loading the driver early in the boot of the operating system. In general, the driver may be configured to detect processes starting on the compute instance, and store a list of such processes as each new process is detected. The driver may also or instead be configured to record each process started and stopped on the compute instance in order to reflect processes that have terminated before a user mode scanner has an opportunity to perform suitable malware analysis.

Loading this driver may include applying the ELAM module or any other suitable early execution techniques. For example, loading the driver may include registering the driver as a kernel-mode driver for execution at an early stage in a boot process. The driver may include a certificate or the like for use by a boot time detection driver of the operating system of the compute instance, e.g., to facilitate accurate identification and registration of the driver during startup. These techniques can facilitate identification and launching of the driver very early in the boot process, e.g., before other, unregistered drivers and other processes such as potential malware.

As shown in step **604**, the method **600** may include detecting processes, e.g., as each new process is launched on the compute instance. This may include any suitable instrumentation to detect new processes. When available, this may use monitoring functions in the kernel. Alternatively, this may include any other suitable instrumentation or the like, e.g., where the corresponding kernel functions are not yet available within the context of the boot and startup of the compute instance.

As shown in step **606**, the method **600** may include storing a list of processes detected by the driver and executing on the compute instance, e.g., by logging such processes in a log **607**. The list of processes may include any suitable information about processes that have been detected such as a name, location, time stamp, and the like. Thus, for example, the driver may record a time stamp indicating a start time for each process identified in the list of processes, and where appropriate, a second time stamp indicating a stop time when the corresponding process terminated.

As shown in step **608**, the method **600** may include launching a first scanner in the user mode. In general, the first scanner may be launched when the user mode is available (e.g., after other prerequisite boot and startup processes have launched or completed), and may be configured to perform a first scan for malware in each process identified in the list of processes when the first scanner launches. It will be noted that the scanning activity such as steps **608**, **610**, and **620** occur in the "user space" on the right side of FIG. **6**, which represents the memory available to the user mode of a compute instance, as distinguished from the kernel space memory which is generally reserved for kernel functions and other system operations.

As shown in step **610**, the method may include requesting logged processes from the log **607** by the first scanner. Where available, this may include a list of processes that have stopped or terminated before the user mode was available, and/or before the first scanner was launched. This usefully prevents early-launching malware from injecting code or otherwise compromising other aspects of the compute instance, and then terminating to prevent detection. It will be noted that the log **607** is depicted between the kernel space and the user space in FIG. **6**. In general, the log **607** may be located in the kernel space, the user space, or some combination of these.

As shown in step **612**, the method **600** may include scanning the logged processes. In one aspect, the first scanner may be configured to asynchronously perform a first scan for malware in each process identified in the list of processes from the driver by scanning at least an executable and an executable file path associated with each process. In this context, the asynchronous execution contemplates execution of the scanning function without gating other program execution on completion of a scan of the processes recorded in the log **607**. Asynchronous execution may also optionally include permitting the processes in the list of processes to continue executing until a scan is completed. While this may present some interstitial risk to the compute

instance, it may also permit important system functions to continue operating during the first scan and can avoid a bottleneck that prevents all user mode activity. In another aspect, the first scanner may execute synchronously in order to ensure a complete scan of all startup processes before other user mode programs can execute. In one aspect, the asynchronous scan may be executed as a single thread.

In general, the first scanner may scan one or more files associated with each process in the list of processes. This may also or instead include scanning related items such as an executable associated with one of the processes, files in an executable path associated with one of the processes, or a dynamic linked library loaded by one of the processes. Because the first scanner is executing in the user mode, any of a wide variety of malware detection techniques may be deployed by the first scanner, including any of the malware detection techniques described herein. It should also be appreciated that, while the first scanner may usefully be applied whenever a compute instance is booted or rebooted, the first scanner may also be useful in other contexts, e.g., if there is a policy change or the like applicable to the compute instance that might alter the scan results.

As shown in step **614**, the method **600** may include remediating malicious code identified in any of the scans described above, e.g., the first scan by first scanner, the second scan by the second scanner, and/or the third scan by the third scanner. For example, the method **600** may include, if malicious code is identified in the one or more files associated with one of the processes in the list of processes in the log **607**, remediating the one of the processes. The remediation may also or instead be directed toward the one or more files associated with the one of the processes in the list of process, e.g., where the process has already terminated as described above, but has been logged by the driver during the boot.

This may include any suitable remediation for malware or other actual or suspected malicious activity. For example, in one aspect, remediating the one of the processes includes quarantining the one of the processes. In another aspect, remediating the one of the processes includes executing the one of the processes in a sandbox or other isolated environment or the like. In another aspect, remediating the one of the processes includes restricting access by the one of the processes to one or more resources of the compute instance such as encrypted files, encryption keys, a network connection, a remote storage facility, a directory, locally connected devices such as removable drives or other peripherals, and so forth. In another aspect, remediating the one of the processes includes executing a malware removal tool for the compute instance to remove the process and/or malware associated with the process. In another aspect, remediating the one of the processes includes executing a malware detection tool for the compute instance, which may include a detection tool for one or more specific malware types or a general malware scanner. It will be understood that remediating the one of the processes may include remediating a specific process or group of related processes, or this may include remediating an entire compute instance, e.g., using any of the techniques described above.

It will be appreciated that a wide range of remediation techniques are known in the art, and may be used instead of, or in addition to, the techniques described above. For example, while code termination may be an important form of intervention, other mitigation techniques may also or instead be used, either alone or in combination with termination of the offending code segment(s). For example, any processes or memory locations causally associated with the code, e.g., by acting on or being acted on by the relevant code, may be labeled as suspicious. These causally related computing objects may also or instead be terminated, deleted, or otherwise remediated. For example, a root cause analysis may be undertaken, e.g., to determine an initial event or root cause of the attack, and to determine related events and associated assets that may have been compromised. Code mitigation may also or instead include quarantine, observation, or the like. In another aspect, the code or associated files or network streams may be colored to indicate suspiciousness, vulnerability, potential compromise, and so forth so that other relevant rules can be applied based on this categorization. For example, in some cases, it may be determined that a vulnerability in an otherwise normal application was exploited in an attack that resulted in the deployment of malware code. Steps may be taken to update or otherwise protect the vulnerable application or to limit or more closely monitor the activity of the application. In another aspect, the code may be moved to a sandbox for execution and examination, or forwarded to a remote malware analysis resource for further review. More generally, any tools or resources for remediating malware, or an affected compute instance, may usefully be deployed upon the detection of actual or potential malware as contemplated herein.

Remediation may also or instead include performing a security update and executing mitigations based on the security update. For example, this may include performing a malware scan, e.g., after waiting the predetermined time period for new security updates. In general, performing the malware scan may include performing signature-based detections for related processes, programs, files, and other computing objects. This may also or instead include malware remediations, e.g., for finding and removing related code, repairing or reinstalling infected programs, and so forth, rolling back registry updates, cleaning up browsers (e.g., to restore settings, clear caches, etc.), updating certificates, and so forth. Where a root cause analysis is performed as described herein, executing mitigations may include remediating the root cause as appropriate.

A history of the compute instance may also or instead be used to detect and remediate malware. For example, where events detected on the compute instance deviate from a baseline of expected activity, any number of responses may be initiated by a local security agent or the like. In one aspect, this may include deployment of known remediations for malicious activity such as quarantine, termination of network communications, termination of processes or applications, an increase in local monitoring activity, messages to a network administrator, filtering of network activity, antivirus scans, deployment of security patches or fixes, and so forth. This may also include policy updates. For example, security policies for compute instances, users, applications, or the like may be updated to security settings that impose stricter controls or limits on activity including, e.g., network activity (bandwidth, data quotas, permitted network addresses, etc.), system changes (e.g., registry entries, certain system calls, etc.), file activity (e.g., changes to file permissions), increased levels of local activity monitoring, and so forth.

Still more generally, any forms of remediation that might usefully follow a malicious activity detection using the scanners described above may usefully be employed in a remediation as described herein.

As shown in step **616**, the method **600** may include launching a second scanner in the user mode. This second

scanner may be configured to detect one or more other processes started after the first scanner is launched.

As shown in step **618**, the method **600** may include detecting and scanning new processes with the second scanner. The second scanner may synchronously perform a second scan for malware in each of the one or more other processes as they are launched, e.g., in the order that they are launched. This synchronous scanning may be multi-threaded, e.g., with a new thread for each new process that is detected, and may be configured to prevent an execution of each of the one or more other processes until a corresponding scan has been completed. Thus, the second scanner may generally detect and check each new process as it is launched. In one aspect, the second scanner may begin to scan only after the first scanner completes a scan of all of the processes identified in the list of processes. If any malware or other malicious or suspicious activity is detected, the method **600** may proceed to step **614** where remediation can be performed.

As shown in step **620**, the method **600** may include launching a third scanner, e.g., in the user mode. The third scanner may be configured to perform a third scan including a background scan of an entire disk associated with the compute instance.

As shown in step **622**, the method **600** may include performing a background scan with the third scanner. In one aspect, this may include a low-priority scan of all available storage for the compute instance, including one or more physical or logical drives and any other memory or storage resources. If any malware or other malicious or suspicious activity is detected, the method **600** may proceed to step **614** where remediation can be performed.

As shown in step **624**, the method **600** may include persisting one or more of the scans. For example, this may include persisting the first scan with a heartbeat or other message to a threat management facility or other remote security resource. The method **600** may also or instead include persisting a scan from at least one of the first scanner and the second scanner, for example, by capturing interim scan results and periodically sending these results with a heartbeat to a threat management facility. In another aspect, the method **600** may include persisting a background scan, e.g., the third scan of steps **620-622**, with a heartbeat to a threat management facility. While step **624** is illustrated as occurring after a remediation step **614**, it will be understood that step **624** may occur prior to, concurrently with, or after any remediation, or any combination of these, or at any other time or combination of times consistent with operation of the method **600** described herein.

More generally, the scans may be persisted in any of a variety of ways. This may include logging directories and files that have been scanned in order to facilitate resuming the scan at a later time. This may also or instead include logging scan results, e.g., with identifiers of suspicious or malicious computing objects. This, or any other suitable interim scan information, may be communicated to a threat management facility in a secure heartbeat or the like, or the interim scan information may be locally stored on a compute instance by a local security agent, with a pointer or identifier to the scan log embedded into a heartbeat to the threat management facility for subsequent location and retrieval of information. More generally, any technique for storing interim scan information, either locally, remotely (e.g., at the threat management facility) or some combination of these, may be used. Similarly, the scan information may be compressed, indexed, encrypted, time stamped, digitally signed, or otherwise processed to improve accessibility, security, reliability, and so forth.

While any scan might usefully be persisted in order to avoid repetition of effort in the event of an unexpected or otherwise sudden shutdown of the compute instance, the background scan may be a slow, low-priority, long-term process extending over minutes, hours, or days in order to prevent excessive use of computing resources that interferes with other user and system functions. In this context, periodic snapshots or the like may be particularly useful to preserve progress that has been made toward a complete scan. When a compute instance is restarted, any persisted scan results may be retrieved from the threat management facility in order to prevent repetition.

Persisting may also be used to fingerprint startup processes. For example, when a compute instance is rebooted, the processes detected by the first scanner may be compared to a record of processes that have been scanned and approved in a prior startup scan. By communicating with a threat management facility or another local or remote resource that stores these prior results, the first scanner may simply refer to this record in lieu of a new scan, subject to changes in the context such as policy changes, reputation changes, or the like that might otherwise alter scan results such as malware detections.

Also disclosed herein are systems for scanning and remediating using the methods described above. In one aspect, a system may include a compute instance, a driver, a first scanner, and a second scanner. The driver may be loaded into and executing in a kernel mode of an operating system for the compute instance before a user mode of the operating system is available, and the driver may be configured to record a list of processes executing on the compute instance by recording processes started on the compute instance after the driver is loaded. The first scanner may be executing in the user mode of the operating system, and configured to perform a first scan for malware in each process identified in the list of processes at a time that the first scanner launches. The second scanner may be executing in the user mode of the operating system, and configured to detect one or more other processes started after the second scanner is launched, to perform a second scan for malware in each of the one or more other processes, and to prevent an execution of each of the one or more other processes until a corresponding scan has been completed.

In another aspect, a system may include a compute instance, a driver, a scanner, and a local security agent. The driver may be loaded into and executing in a kernel mode of an operating system for the compute instance before a user mode of the operating system is available, and the driver may be configured to record a list of processes including each process started and stopped on the compute instance after the driver is loaded. The scanner may be executing in the user mode of the operating system, and configured to perform a scan for malware in each process identified in the list of processes at a time that the scanner launches. The local security agent may be executing on the compute instance (e.g., in the user mode) and configured to remediate malicious code identified by the scanner.

In general, these systems may also be configured to remediate the compute instance as generally described herein, and/or otherwise perform any of the steps described above.

The above systems, devices, methods, processes, and the like may be realized in hardware, software, or any combination of these suitable for a particular application. The

hardware may include a general-purpose computer and/or dedicated computing device. This includes realization in one or more microprocessors, microcontrollers, embedded microcontrollers, programmable digital signal processors or other programmable devices or processing circuitry, along with internal and/or external memory. This may also, or instead, include one or more application specific integrated circuits, programmable gate arrays, programmable array logic components, or any other device or devices that may be configured to process electronic signals. It will further be appreciated that a realization of the processes or devices described above may include computer-executable code created using a structured programming language such as C, an object oriented programming language such as C++, or any other high-level or low-level programming language (including assembly languages, hardware description languages, and database programming languages and technologies) that may be stored, compiled or interpreted to run on one of the above devices, as well as heterogeneous combinations of processors, processor architectures, or combinations of different hardware and software. In another aspect, the methods may be embodied in systems that perform the steps thereof, and may be distributed across devices in a number of ways. At the same time, processing may be distributed across devices such as the various systems described above, or all of the functionality may be integrated into a dedicated, standalone device or other hardware. In another aspect, means for performing the steps associated with the processes described above may include any of the hardware and/or software described above. All such permutations and combinations are intended to fall within the scope of the present disclosure.

Embodiments disclosed herein may include computer program products comprising computer-executable code or computer-usable code that, when executing on one or more computing devices, performs any and/or all of the steps thereof. The code may be stored in a non-transitory fashion in a computer memory, which may be a memory from which the program executes (such as random-access memory associated with a processor), or a storage device such as a disk drive, flash memory or any other optical, electromagnetic, magnetic, infrared or other device or combination of devices. In another aspect, any of the systems and methods described above may be embodied in any suitable transmission or propagation medium carrying computer-executable code and/or any inputs or outputs from same.

The method steps of the implementations described herein are intended to include any suitable method of causing such method steps to be performed, consistent with the patentability of the following claims, unless a different meaning is expressly provided or otherwise clear from the context. So, for example, performing the step of X includes any suitable method for causing another party such as a remote user, a remote processing resource (e.g., a server or cloud computer) or a machine to perform the step of X. Similarly, performing steps X, Y and Z may include any method of directing or controlling any combination of such other individuals or resources to perform steps X, Y and Z to obtain the benefit of such steps. Thus, method steps of the implementations described herein are intended to include any suitable method of causing one or more other parties or entities to perform the steps, consistent with the patentability of the following claims, unless a different meaning is expressly provided or otherwise clear from the context. Such parties or entities need not be under the direction or control of any other party or entity, and need not be located within a particular jurisdiction.

It will be appreciated that the methods and systems described above are set forth by way of example and not of limitation. Numerous variations, additions, omissions, and other modifications will be apparent to one of ordinary skill in the art. In addition, the order or presentation of method steps in the description and drawings above is not intended to require this order of performing the recited steps unless a particular order is expressly required or otherwise clear from the context. Thus, while particular embodiments have been shown and described, it will be apparent to those skilled in the art that various changes and modifications in form and details may be made therein without departing from the spirit and scope of this disclosure and are intended to form a part of the invention as defined by the following claims, which are to be interpreted in the broadest sense allowable by law.

What is claimed is:

**1**. A computer program product comprising computer executable code embodied in a non-transitory computer readable medium that, when executing on a computing device, performs the steps of:

loading a driver in a kernel mode of an operating system during a boot of the operating system on a compute instance before a user mode of the operating system is available, the driver configured to detect processes starting on the compute instance;

storing a list of processes detected by the driver and executing on the compute instance;

launching a first scanner in the user mode, the first scanner configured to asynchronously perform a first scan for malware in each process identified in the list of processes from the driver by scanning at least an executable and an executable file path associated with each process;

launching a second scanner in the user mode, the second scanner configured to detect one or more other processes started after the first scanner is launched, to synchronously perform a second scan for malware in each of the one or more other processes, and to prevent an execution of each of the one or more other processes until a corresponding scan has been completed; and

remediating malicious code identified in at least one of the first scan and the second scan.

**2**. The computer program product of claim **1** wherein the second scanner begins to scan after the first scanner completes a scan of all of the processes identified in the list of processes.

**3**. The computer program product of claim **1** wherein the first scan includes a scan of at least one of an executable associated with one of the processes, files in an executable path associated with one of the processes, or a dynamic linked library loaded by one of the processes.

**4**. The computer program product of claim **1** wherein loading the driver includes registering the driver as a kernel-mode driver for execution at an early stage in a boot process.

**5**. The computer program product of claim **1** wherein the driver records a time stamp indicating a start time for each process identified in the list of processes.

**6**. The computer program product of claim **1** further comprising computer executable code that performs the step of launching a third scanner configured to perform a third scan including a background scan of an entire disk associated with the compute instance.

**7**. The computer program product of claim **6** further comprising computer executable code that performs the step

of persisting at least one of the first scan, the second scan, and the third scan with a heartbeat to a threat management facility.

**8**. A method comprising:

loading a driver during a boot of an operating system on a compute instance, the driver loaded before a user mode of the operating system is available and the driver configured to store a list of processes executing on the compute instance;

launching a first scanner in the user mode, the first scanner configured to perform a first scan for malware in each process identified in the list of processes when the first scanner launches; and

launching a second scanner in the user mode, the second scanner configured to detect one or more other processes started after the first scanner is launched, to perform a second scan for malware in each of the one or more other processes, and to prevent an execution of each of the one or more other processes until a corresponding scan has been completed.

**9**. The method of claim **8** wherein the second scanner synchronously scans the one or more other processes in an order that the one or more other processes launched.

**10**. The method of claim **8** wherein the second scanner begins to scan after the first scanner completes a scan of all processes identified in the list of processes.

**11**. The method of claim **8** wherein the first scanner asynchronously scans processes identified in the list of processes.

**12**. The method of claim **8** wherein the first scan includes a scan of at least one of an executable associated with a process in the list of processes, files in an executable path associated with a process in the list of processes, or a dynamic linked library loaded by a process in the list of processes.

**13**. The method of claim **8** wherein loading the driver includes loading the driver early in the boot of the operating system.

**14**. The method of claim **8** wherein loading the driver includes registering the driver as a kernel-mode driver for execution at an early stage in a boot process.

**15**. The method of claim **8** wherein the driver includes a certificate for use by a boot time detection driver of the operating system of the compute instance.

**16**. The method of claim **8** wherein the driver records a time stamp indicating a start time for each process identified in the list of processes.

**17**. The method of claim **8** further comprising persisting the first scan with a heartbeat to a threat management facility.

**18**. The method of claim **8** further comprising launching a third scanner, the third scanner configured to perform a background scan of an entire disk associated with the compute instance.

**19**. The method of claim **18** further comprising persisting the background scan with a heartbeat to a threat management facility.

**20**. A system comprising:

a compute instance;

a driver loaded into and executing in a kernel mode of an operating system for the compute instance before a user mode of the operating system is available, the driver configured to record a list of processes executing on the compute instance by recording processes started on the compute instance after the driver is loaded;

a first scanner executing in the user mode of the operating system, the first scanner configured to perform a first scan for malware in each process identified in the list of processes at a time that the first scanner launches; and

a second scanner executing in the user mode of the operating system, the second scanner configured to detect one or more other processes started after the second scanner is launched, to perform a second scan for malware in each of the one or more other processes, and to prevent an execution of each of the one or more other processes until a corresponding scan has been completed.

* * * * *